

QTrk

Generated by Doxygen 1.8.11

Contents

1	Queued Tracker software	1
1.1	Introduction	1
1.2	History	1
1.3	Implementation	1
1.4	Required software	2
1.5	Credits	2
2	Todo List	3
3	Module Index	5
3.1	Modules	5
4	Namespace Index	7
4.1	Namespace List	7
5	Hierarchical Index	9
5.1	Class Hierarchy	9
6	Class Index	13
6.1	Class List	13
7	File Index	17
7.1	File List	17

8 Module Documentation	19
8.1 LabVIEW datatypes and helper functions	19
8.1.1 Detailed Description	20
8.1.2 Typedef Documentation	20
8.1.2.1 ppFloatArray	20
8.1.2.2 ppFloatArray2	20
8.1.3 Function Documentation	20
8.1.3.1 ArgumentErrorMsg(ErrorCluster *e, const std::string &msg)	20
8.1.3.2 FillErrorCluster(MgErr err, const char *message, ErrorCluster *error)	20
8.1.3.3 LVGetStringArray(int count, LStrHandle *str)	21
8.1.3.4 ResizeLVArray(LVArrayND< T, N > **&d, int *dims)	21
8.1.3.5 ResizeLVArray(LVArray< T > **&d, int elems)	21
8.1.3.6 ResizeLVArray2D(LVArray2D< T > **&d, int rows, int cols)	21
8.1.3.7 ResizeLVArray3D(LVArray3D< T > **&d, int depth, int rows, int cols)	21
8.1.3.8 SetLVString(LStrHandle str, const char *text)	22
8.1.3.9 ValidateTracker(QueuedTracker *tracker, ErrorCluster *e, const char *funcname)	22
8.2 API - LabVIEW	23
8.2.1 Detailed Description	23
8.2.2 Enumeration Type Documentation	24
8.2.2.1 QueueFrameFlags	24
8.2.3 Function Documentation	24
8.2.3.1 qtrk_clear_results(QueuedTracker *qtrk, ErrorCluster *e)	24
8.2.3.2 qtrk_create(QTrkSettings *settings, LStrHandle warnings, ErrorCluster *e)	24
8.2.3.3 qtrk_destroy(QueuedTracker *qtrk, ErrorCluster *error)	24
8.2.3.4 qtrk_dump_memleaks()	25
8.2.3.5 qtrk_flush(QueuedTracker *qtrk, ErrorCluster *e)	25
8.2.3.6 qtrk_generate_image_from_lut(LVArray2D< float > **image, LVArray2D< float > **lut, float *LUTradii, vector2f *position, float z, float M, float sigma_noise)	25
8.2.3.7 qtrk_get_profile_report(QueuedTracker *qtrk, LStrHandle str)	25
8.2.3.8 qtrk_get_results(QueuedTracker *qtrk, LocalizationResult *results, int maxResults, int sortByID, ErrorCluster *e)	25

8.2.3.9	qtrk_get_ZLUT(QueuedTracker *tracker, LVArray3D< float > **pzlut, ErrorCluster *e)	26
8.2.3.10	qtrk_idle(QueuedTracker *qtrk, ErrorCluster *e)	26
8.2.3.11	qtrk_queue_array(QueuedTracker *qtrk, ErrorCluster *error, LVArray2D< uchar > **data, uint pdt, const LocalizationJob *jobInfo)	26
8.2.3.12	qtrk_queue_float(QueuedTracker *qtrk, ErrorCluster *error, LVArray2D< float > **data, const LocalizationJob *jobInfo)	26
8.2.3.13	qtrk_queue_frame(QueuedTracker *qtrk, uchar *image, int pitch, int w, int h, uint pdt, ROIPosition *pos, int numROI, const LocalizationJob *pJobInfo, QueueFrameFlags flags, ErrorCluster *e)	26
8.2.3.14	qtrk_queue_pitchedmem(QueuedTracker *qtrk, uchar *data, int pitch, uint pdt, const LocalizationJob *jobInfo)	27
8.2.3.15	qtrk_queue_u16(QueuedTracker *qtrk, ErrorCluster *error, LVArray2D< ushort > **data, const LocalizationJob *jobInfo)	27
8.2.3.16	qtrk_queue_u8(QueuedTracker *qtrk, ErrorCluster *error, LVArray2D< uchar > **data, const LocalizationJob *jobInfo)	27
8.2.3.17	qtrk_read_timestamp(uchar *image, int w, int h)	27
8.2.3.18	qtrk_resultcount(QueuedTracker *qtrk, ErrorCluster *e)	27
8.2.3.19	qtrk_set_ZLUT(QueuedTracker *tracker, LVArray3D< float > **pZlut, LVArray< float > **zcmpWindow, int normalize, ErrorCluster *e)	28
8.2.3.20	qtrkcuda_device_count(ErrorCluster *e)	28
8.2.3.21	qtrkcuda_set_device_list(LVArray< int > **devices)	28
8.3	API - C	29
8.3.1	Detailed Description	29
8.3.2	Function Documentation	29
8.3.2.1	QTrkBuildLUT(QueuedTracker *qtrk, void *data, int pitch, QTRK_PixelDataType pdt, bool imageLUT, int plane)	29
8.3.2.2	QTrkClearResults(QueuedTracker *qtrk)	30
8.3.2.3	QTrkCreateInstance(QTrkSettings *cfg)	30
8.3.2.4	QTrkFetchResults(QueuedTracker *qtrk, LocalizationResult *results, int maxResults)	30
8.3.2.5	QTrkFinalizeLUT(QueuedTracker *qtrk)	30
8.3.2.6	QTrkFlush(QueuedTracker *qtrk)	30
8.3.2.7	QTrkFreeInstance(QueuedTracker *qtrk)	30
8.3.2.8	QTrkGetComputedConfig(QueuedTracker *qtrk, QTrkComputedConfig *cfg)	31

8.3.2.9	QTrkGetProfileReport(QueuedTracker *qtrk, char *dst, int maxStrLen)	31
8.3.2.10	QTrkGetQueueLength(QueuedTracker *qtrk, int *maxQueueLen)	31
8.3.2.11	QTrkGetRadialZLUT(QueuedTracker *qtrk, float *dst)	31
8.3.2.12	QTrkGetRadialZLUTSize(QueuedTracker *qtrk, int *count, int *planes, int *radialsteps)	31
8.3.2.13	QTrkGetResultCount(QueuedTracker *qtrk)	31
8.3.2.14	QTrkGetWarnings(QueuedTracker *qtrk, char *dst, int maxStrLen)	31
8.3.2.15	QTrkIsIdle(QueuedTracker *qtrk)	32
8.3.2.16	QTrkScheduleFrame(QueuedTracker *qtrk, void *imgptr, int pitch, int width, int height, ROIPosition *positions, int numROI, QTRK_PixelDataType pdt, const LocalizationJob *jobInfo)	32
8.3.2.17	QTrkScheduleLocalization(QueuedTracker *qtrk, void *data, int pitch, QTRK_PixelDataType pdt, const LocalizationJob *jobInfo)	32
8.3.2.18	QTrkSetLocalizationMode(QueuedTracker *qtrk, LocMode_t locType)	32
8.3.2.19	QTrkSetRadialZLUT(QueuedTracker *qtrk, float *data, int count, int planes, float *zcmp=0)	32
9	Namespace Documentation	33
9.1	BeadFinder Namespace Reference	33
9.1.1	Function Documentation	33
9.1.1.1	Find(ImageData *img, float *sample, Config *cfg)	33
9.1.1.2	Find(uint8_t *img, int pitch, int w, int h, int smpCornerX, int smpCornerY, Config *cfg)	34
9.2	kissfft_utils Namespace Reference	34
9.3	qtrk Namespace Reference	34
9.4	sfft Namespace Reference	35
9.4.1	Function Documentation	35
9.4.1.1	fft(size_t N, std::complex< T > *zs, std::complex< T > *twiddles)	35
9.4.1.2	fft_forward(size_t N, std::complex< T > *zs, std::complex< T > *twiddles)	35
9.4.1.3	fft_inverse(size_t N, std::complex< T > *zs, std::complex< T > *twiddles)	36
9.4.1.4	fill_twiddles(int N)	36
9.4.1.5	swap(T &a, T &b)	36

10 Class Documentation	37
10.1 Atomic< T > Class Template Reference	37
10.1.1 Constructor & Destructor Documentation	37
10.1.1.1 Atomic(const T &o=T())	37
10.1.2 Member Function Documentation	37
10.1.2.1 get() const	37
10.1.2.2 operator T() const	38
10.1.2.3 operator=(const T &x)	38
10.1.2.4 set(const T &x)	38
10.1.3 Member Data Documentation	38
10.1.3.1 data	38
10.1.3.2 m	38
10.2 BaseKernelParams Struct Reference	38
10.2.1 Member Data Documentation	38
10.2.1.1 images	38
10.2.1.2 imgmeans	38
10.2.1.3 locParams	38
10.2.1.4 njobs	38
10.3 BenchmarkLUT Class Reference	39
10.3.1 Constructor & Destructor Documentation	39
10.3.1.1 BenchmarkLUT()	39
10.3.1.2 BenchmarkLUT(ImageData *lut)	39
10.3.1.3 BenchmarkLUT(const char *file)	39
10.3.2 Member Function Documentation	40
10.3.2.1 CleanupLUT(ImageData &lut)	40
10.3.2.2 GenerateLUT(ImageData *lut)	40
10.3.2.3 GenerateSample(ImageData *image, vector3f pos, float minRadius, float maxRadius)	40
10.3.2.4 Load(ImageData *lut)	40
10.3.2.5 Load(const char *file)	41
10.3.3 Member Data Documentation	41

10.3.3.1 lut_h	41
10.3.3.2 lut_w	41
10.3.3.3 max_a	41
10.3.3.4 max_b	41
10.3.3.5 max_c	41
10.3.3.6 normprof	41
10.4 BinaryResultFile Class Reference	41
10.4.1 Constructor & Destructor Documentation	42
10.4.1.1 BinaryResultFile(const char *fn, bool write)	42
10.4.2 Member Function Documentation	42
10.4.2.1 LoadRow(std::vector< vector3f > &pos)	42
10.4.2.2 SaveRow(std::vector< vector3f > &pos)	42
10.4.3 Member Data Documentation	42
10.4.3.1 f	42
10.5 ClImageData Class Reference	43
10.5.1 Constructor & Destructor Documentation	43
10.5.1.1 ClImageData(int w, int h)	43
10.5.1.2 ClImageData(const ClImageData &other)	43
10.5.1.3 ClImageData()	43
10.5.1.4 ~ClImageData()	43
10.5.1.5 ClImageData(const TlImageData< float > &src)	44
10.5.2 Member Function Documentation	44
10.5.2.1 operator=(const ClImageData &src)	44
10.5.2.2 operator=(const TlImageData< float > &src)	44
10.6 LsqSqQuadFit< T >::Coeff Struct Reference	44
10.6.1 Member Function Documentation	45
10.6.1.1 abc(T &a, T &b, T &c, T &d)	45
10.6.2 Member Data Documentation	45
10.6.2.1 s00	45
10.6.2.2 s01	45

10.6.2.3	s10	45
10.6.2.4	s11	45
10.6.2.5	s20	45
10.6.2.6	s21	45
10.6.2.7	s30	45
10.6.2.8	s40	45
10.7	sfft::complex< T > Struct Template Reference	45
10.7.1	Constructor & Destructor Documentation	46
10.7.1.1	complex()	46
10.7.1.2	complex(T a, T b=0.0)	46
10.7.1.3	complex(const std::complex< T > &a)	46
10.7.2	Member Function Documentation	46
10.7.2.1	conjugate()	46
10.7.2.2	imag()	46
10.7.2.3	imag() const	47
10.7.2.4	operator*(const T &b) const	47
10.7.2.5	operator*(const complex &b) const	47
10.7.2.6	operator*=(const T &b)	47
10.7.2.7	operator+(const complex &b) const	47
10.7.2.8	operator+=(const complex &b)	47
10.7.2.9	operator-(const complex &b) const	47
10.7.2.10	real()	47
10.7.2.11	real() const	47
10.7.3	Member Data Documentation	48
10.7.3.1	x	48
10.7.3.2	y	48
10.8	ComputeMaxInterp< T, numPts > Class Template Reference	48
10.8.1	Member Function Documentation	48
10.8.1.1	Compute(T *data, int len, const T *weights, LsqSqQuadFit< T > *fit=0)	48
10.8.1.2	max_(T a, T b)	49

10.8.1.3 min_(T a, T b)	49
10.9 BeadFinder::Config Struct Reference	49
10.9.1 Member Function Documentation	49
10.9.1.1 MinPixelDistance()	49
10.9.2 Member Data Documentation	49
10.9.2.1 img_distance	49
10.9.2.2 roi	49
10.9.2.3 similarity	49
10.10 CPUTracker Class Reference	50
10.10.1 Member Enumeration Documentation	51
10.10.1.1 LUTProfileMaxComputeMode	51
10.10.2 Constructor & Destructor Documentation	52
10.10.2.1 CPUTracker(int w, int h, int xcorwindow=128, bool testRun=false)	52
10.10.2.2 ~CPUTracker()	52
10.10.3 Member Function Documentation	52
10.10.3.1 AllocateQIFFTs(int nsteps)	52
10.10.3.2 ApplyOffsetGain(float *offset, float *gain, float offsetFactor, float gainFactor)	53
10.10.3.3 CalculateErrorCurve(double *errorcurve_dest, float *profile, float *zlut_sel)	53
10.10.3.4 CalculateErrorFlag(double *prof1, double *prof2)	53
10.10.3.5 CalculateInterpolatedZLUTProfile(float *profile_dest, float z, int zlutIndex)	54
10.10.3.6 CheckBoundaries(vector2f center, float radius)	54
10.10.3.7 Compute2DGaussianMLE(vector2f initial, int iterations, float sigma)	54
10.10.3.8 ComputeAsymmetry(vector2f center, int radialSteps, int angularSteps, float minRadius, float maxRadius, float *dstAngProf=0)	55
10.10.3.9 ComputeMeanAndCOM(float bgcorrection=0.0f)	55
10.10.3.10 ComputeQI(vector2f initial, int iterations, int radialSteps, int angularStepsPerQuadrant, float angStepIterationFactor, float minRadius, float maxRadius, bool &boundaryHit, float *radialweights=0)	56
10.10.3.11 ComputeQuadrantProfile(scalar_t *dst, int radialSteps, int angularSteps, int quadrant, float minRadius, float maxRadius, vector2f center, float *radialWeights=0)	57
10.10.3.12 ComputeRadialProfile(float *dst, int radialSteps, int angularSteps, float minradius, float maxradius, vector2f center, bool crp, bool *boundaryHit=0, bool normalize=true)	58

10.10.3.13ComputeXCorInterpolated(vector2f initial, int iterations, int profileWidth, bool &boundaryHit)	58
10.10.3.14ComputeZ(vector2f center, int angularSteps, int zlutIndex, bool *boundaryHit=0, float *profile=0, float *cmpprof=0, bool normalizeProfile=true)	59
10.10.3.15FourierRadialProfile(float *dst, int radialSteps, int angularSteps, float minradius, float maxradius)	59
10.10.3.16FourierTransform2D()	60
10.10.3.17GetDebugImage()	60
10.10.3.18GetHeight()	60
10.10.3.19GetPixel(int x, int y)	60
10.10.3.20GetRadialZLUT(int index)	60
10.10.3.21GetWidth()	60
10.10.3.22KeepInsideBoundaries(vector2f *center, float radius)	61
10.10.3.23LUTProfileCompare(float *profile, int zlutIndex, float *cmpProf, LUTProfile ← MaxComputeMode maxPosMethod, float *lsqfittedcurve=0, int *maxPos=0, int frameNum=0)	61
10.10.3.24LUTProfileCompareAdjustedWeights(float *rprof, int zlutIndex, float z_estim)	63
10.10.3.25Normalize(float *image=0)	63
10.10.3.26QI_ComputeOffset(complex_t *qi_profile, int nr, int axisForDebug)	63
10.10.3.27QuadrantAlign(vector3f initial, int beadIndex, int angularStepsPerQuadrant, bool &boundaryHit)	64
10.10.3.28QuadrantAlign_ComputeOffset(complex_t *profile, complex_t *zlut_prof_fft, int nr, int axisForDebug)	65
10.10.3.29SaveImage(const char *filename)	65
10.10.3.30SetImage(TPixel *srclImage, uint srcpitch)	65
10.10.3.31SetImage16Bit(ushort *srclImage, uint srcpitch)	65
10.10.3.32SetImage8Bit(uchar *srclImage, uint srcpitch)	66
10.10.3.33SetImageFloat(float *srclImage)	66
10.10.3.34SetRadialWeights(float *w)	66
10.10.3.35SetRadialZLUT(float *data, int planes, int res, int num_zluts, float minradius, float maxradius, bool copyMemory, bool useCorrelation)	66
10.10.4 Member Data Documentation	66
10.10.4.1 debugImage	66
10.10.4.2 fft2d	66

10.10.4.3 height	66
10.10.4.4 mean	67
10.10.4.5 qa_fft_backward	67
10.10.4.6 qa_fft_forward	67
10.10.4.7 qi_fft_backward	67
10.10.4.8 qi_fft_forward	67
10.10.4.9 qi_radialsteps	67
10.10.4.10quadrantDirs	67
10.10.4.11radialDirs	67
10.10.4.12rclImage	67
10.10.4.13stdev	67
10.10.4.14testRun	67
10.10.4.15trackerID	67
10.10.4.16width	67
10.10.4.17xcorBuffer	67
10.10.4.18xcorw	67
10.10.4.19zlut_count	67
10.10.4.20zlut_maxradius	67
10.10.4.21zlut_memoryOwner	67
10.10.4.22zlut_minradius	67
10.10.4.23zlut_planes	67
10.10.4.24zlut_radialweights	67
10.10.4.25zlut_res	67
10.10.4.26zlut_useCorrelation	67
10.10.4.27zluts	67
10.11CUDADeviceInfo Struct Reference	68
10.11.1 Member Data Documentation	68
10.11.1.1 clockRate	68
10.11.1.2 major	68
10.11.1.3 minor	68

10.11.1.4 multiProcCount	68
10.11.1.5 name	68
10.12 <code>cudaImageList< T ></code> Struct Template Reference	68
10.12.1 Member Enumeration Documentation	69
10.12.1.1 anonymous enum	69
10.12.2 Member Function Documentation	70
10.12.2.1 alloc(int w, int h, int amount)	70
10.12.2.2 allocateHostImageBuffer(pinned_array< T, Flags > &hostImgBuf)	70
10.12.2.3 bind(texture< T, cudaTextureType2D, cudaMemcpyMode> &texref)	70
10.12.2.4 boundaryHit(float2 center, float radius)	70
10.12.2.5 capacity()	70
10.12.2.6 clear()	70
10.12.2.7 computeImagePos(int &x, int &y, int idx)	71
10.12.2.8 copyImageToDevice(int img, T *src, bool async=false, cudaStream_t s=0)	71
10.12.2.9 copyImageToHost(int img, T *dst, bool async=false, cudaStream_t s=0)	71
10.12.2.10copyToDevice(T *src, bool async=false, cudaStream_t s=0)	71
10.12.2.11copyToDevice(T *src, int numImages, bool async=false, cudaStream_t s=0)	71
10.12.2.12copyToHost(T *dst, bool async=false, cudaStream_t s=0)	72
10.12.2.13emptyList()	72
10.12.2.14free()	72
10.12.2.15fullheight()	72
10.12.2.16fullwidth()	72
10.12.2.17get(int i)	72
10.12.2.18interp(T a, T b, float x)	72
10.12.2.19interpolate(float x, float y, int idx, bool &outside)	73
10.12.2.20interpolateFromTexture(texture< T, cudaTextureType2D, cudaMemcpyMode> texref, float x, float y, int idx, bool &outside)	73
10.12.2.21isEmpty()	73
10.12.2.22numpixels()	73
10.12.2.23pixel(int x, int y, int imgIndex)	73
10.12.2.24pixel_oobcheck(int x, int y, int imgIndex, T border=0.0f)	74

10.12.2.25pixelAddress(int x, int y, int imgIndex)	74
10.12.2.26totalNumBytes()	74
10.12.2.27totalNumPixels()	74
10.12.2.28unbind(texture< T, cudaTextureType2D, cudaReadModeElementType > &texref)	74
10.12.3 Member Data Documentation	74
10.12.3.1 count	74
10.12.3.2 data	74
10.12.3.3 h	74
10.12.3.4 pitch	74
10.12.3.5 w	74
10.13QueuedCUDATracker::Device Struct Reference	75
10.13.1 Constructor & Destructor Documentation	75
10.13.1.1 Device(int index)	75
10.13.1.2 ~Device()	75
10.13.2 Member Function Documentation	76
10.13.2.1 SetPixelCalibrationImages(float *offset, float *gain, int img_width, int img_height)	76
10.13.2.2 SetRadialWeights(float *zcmp)	76
10.13.2.3 SetRadialZLUT(float *data, int radialsteps, int planes, int numLUTs)	76
10.13.3 Member Data Documentation	76
10.13.3.1 calib_gain	76
10.13.3.2 calib_offset	76
10.13.3.3 index	77
10.13.3.4 qalign_instance	77
10.13.3.5 qi_instance	77
10.13.3.6 radial_zlut	77
10.13.3.7 zcompareWindow	77
10.13.3.8 zlut_trigtable	77
10.14device_vec< T > Class Template Reference	77
10.14.1 Constructor & Destructor Documentation	78
10.14.1.1 device_vec()	78

10.14.1.2 <code>device_vec(size_t N)</code>	78
10.14.1.3 <code>device_vec(const device_vec< T > &src)</code>	78
10.14.1.4 <code>device_vec(const std::vector< T > &src)</code>	78
10.14.1.5 <code>~device_vec()</code>	78
10.14.2 Member Function Documentation	78
10.14.2.1 <code>copyToDevice(const std::vector< T > &src, bool async=false, cudaStream_t s=0)</code>	78
10.14.2.2 <code>copyToDevice(const T *first, size_t size, bool async=false, cudaStream_t s=0)</code>	79
10.14.2.3 <code>copyToHost(T *dst, bool async, cudaStream_t s=0)</code>	79
10.14.2.4 <code>copyToHost(std::vector< T > &dst, bool async, cudaStream_t s=0)</code>	79
10.14.2.5 <code>free()</code>	79
10.14.2.6 <code>init(size_t s)</code>	79
10.14.2.7 <code>memsize()</code>	80
10.14.2.8 <code>operator std::vector< T >() const</code>	80
10.14.2.9 <code>operator=(const std::vector< T > &src)</code>	80
10.14.2.10 <code>operator=(const device_vec< T > &src)</code>	80
10.14.2.11 <code>toVector()</code>	80
10.14.3 Member Data Documentation	80
10.14.3.1 <code>data</code>	80
10.14.3.2 <code>size</code>	80
10.15 <code>QI::DeviceInstance</code> Struct Reference	81
10.15.1 Constructor & Destructor Documentation	81
10.15.1.1 <code>~DeviceInstance()</code>	81
10.15.1.2 <code>DeviceInstance()</code>	81
10.15.2 Member Data Documentation	81
10.15.2.1 <code>d_qiparams</code>	81
10.15.2.2 <code>d_radialweights</code>	81
10.15.2.3 <code>qi_trigtable</code>	81
10.16 <code>ErrorCluster</code> Struct Reference	81
10.16.1 Member Data Documentation	82
10.16.1.1 <code>code</code>	82

10.16.1.2 message	82
10.16.1.3 status	82
10.17CPUTracker::FFT2D Class Reference	82
10.17.1 Constructor & Destructor Documentation	82
10.17.1.1 FFT2D(int w, int h)	82
10.17.1.2 ~FFT2D()	82
10.17.2 Member Function Documentation	83
10.17.2.1 Apply(float *d)	83
10.17.3 Member Data Documentation	83
10.17.3.1 cbuf	83
10.17.3.2 xfft	83
10.17.3.3 yfft	83
10.18FFT2DTracker Class Reference	83
10.18.1 Constructor & Destructor Documentation	84
10.18.1.1 FFT2DTracker(int w, int h)	84
10.18.1.2 ~FFT2DTracker()	84
10.18.2 Member Function Documentation	84
10.18.2.1 ComputeMax2DInterpolated(float *img)	84
10.18.2.2 ComputeXCor(float *image)	84
10.18.2.3 GetAutoConvResults()	84
10.18.3 Member Data Documentation	84
10.18.3.1 fft_buf	84
10.18.3.2 fft_buf_mirrored	84
10.18.3.3 height	84
10.18.3.4 mirror2D	84
10.18.3.5 plan_bw2D	84
10.18.3.6 plan_fw2D	84
10.18.3.7 width	84
10.19ResultManager::FrameCounters Struct Reference	85
10.19.1 Constructor & Destructor Documentation	85

10.19.1.1 FrameCounters()	85
10.19.2 Member Data Documentation	85
10.19.2.1 capturedFrames	85
10.19.2.2 fileError	85
10.19.2.3 lastSaveFrame	85
10.19.2.4 localizationsDone	85
10.19.2.5 lostFrames	85
10.19.2.6 processedFrames	85
10.19.2.7 startFrame	85
10.20ResultManager::FrameResult Struct Reference	86
10.20.1 Constructor & Destructor Documentation	86
10.20.1.1 FrameResult(int nResult, int nFrameInfo)	86
10.20.2 Member Data Documentation	86
10.20.2.1 count	86
10.20.2.2 frameInfo	86
10.20.2.3 hasFrameInfo	86
10.20.2.4 results	86
10.20.2.5 timestamp	86
10.21CPUTracker::Gauss2DResult Struct Reference	86
10.21.1 Member Data Documentation	87
10.21.1.1 bg	87
10.21.1.2 l0	87
10.21.1.3 pos	87
10.22Threads::Handle Struct Reference	87
10.22.1 Member Data Documentation	87
10.22.1.1 callback	87
10.22.1.2 param	87
10.22.1.3 threadID	87
10.22.1.4 winhdl	87
10.23qtrk::hash_map< TKey, T > Class Template Reference	88

10.24qtrk::hash_set< T > Class Template Reference	88
10.25Image4DCudaArray< T > Struct Template Reference	88
10.25.1 Constructor & Destructor Documentation	89
10.25.1.1 Image4DCudaArray(int sx, int sy, int numImg, int sL)	89
10.25.1.2 ~Image4DCudaArray()	90
10.25.2 Member Function Documentation	90
10.25.2.1 bind(texture< T, cudaTextureType2DLayered, cudaMemcpyModeElementType > &texref)	90
10.25.2.2 bind(surface< void, cudaSurfaceType2DLayered > &surf)	90
10.25.2.3 clear()	90
10.25.2.4 copyImageToDevice(int img, int layer, T *src, bool async=false, cudaStream_t s=0)	91
10.25.2.5 copyImageToHost(int img, int layer, T *dst, bool async=false, cudaStream_t s=0)	91
10.25.2.6 copyToDevice(T *src, bool async=false, cudaStream_t s=0)	91
10.25.2.7 copyToHost(T *dst, bool async=false, cudaStream_t s=0)	92
10.25.2.8 free()	92
10.25.2.9 getExtent()	92
10.25.2.10getImagePos(int image)	92
10.25.2.11kernelInst()	92
10.25.2.12unbind(texture< T, cudaTextureType2DLayered, cudaMemcpyModeElementType > &texref)	92
10.25.3 Member Data Documentation	93
10.25.3.1 array	93
10.25.3.2 imgh	93
10.25.3.3 imgw	93
10.25.3.4 layerh	93
10.25.3.5 layerw	93
10.25.3.6 nlayers	93
10.25.3.7 numImg	93
10.26Image4DMemory< T > Class Template Reference	93
10.26.1 Constructor & Destructor Documentation	94
10.26.1.1 Image4DMemory(int w, int h, int d, int L)	94

10.26.1.2 ~Image4DMemory()	94
10.26.2 Member Function Documentation	94
10.26.2.1 bind()	94
10.26.2.2 clear()	94
10.26.2.3 copyImageToDevice(int z, int l, T *src, bool async=false, cudaStream_t s=0)	94
10.26.2.4 copyImageToHost(int z, int l, T *dst, bool async=false, cudaStream_t s=0)	95
10.26.2.5 copyToDevice(T *src, bool async=false, cudaStream_t s=0)	95
10.26.2.6 copyToHost(T *dst, bool async=false, cudaStream_t s=0)	95
10.26.2.7 free()	95
10.26.2.8 getImgAddr(int2 imgpos)	95
10.26.2.9 read(const KernelParams &kp, int x, int y, int2 imgpos)	95
10.26.2.10unbind()	96
10.26.2.11write(T value, const KernelParams &kp, int x, int y, int2 imgpos)	96
10.26.3 Member Data Documentation	96
10.26.3.1 kp	96
10.26.3.2 layers	96
10.26.3.3 rows	96
10.26.3.4 totallImg	96
10.27 ImageLUTConfig Struct Reference	96
10.27.1 Member Function Documentation	97
10.27.1.1 empty()	97
10.27.2 Member Data Documentation	97
10.27.2.1 h	97
10.27.2.2 nLUTs	97
10.27.2.3 planes	97
10.27.2.4 w	97
10.27.2.5 xscale	97
10.27.2.6 yscale	97
10.28 ImageSampler_InterpolatedTexture Class Reference	97
10.28.1 Member Function Documentation	97

10.28.1.1 BindTexture(cudalImageListf &images)	97
10.28.1.2 Index(cudalImageListf &images, int x, int y, int img)	98
10.28.1.3 Interpolated(cudalImageListf &images, float x, float y, int img, bool &outside)	98
10.28.1.4 UnbindTexture(cudalImageListf &images)	98
10.29 ImageSampler_MemCopy Class Reference	98
10.29.1 Member Function Documentation	98
10.29.1.1 BindTexture(cudalImageListf &images)	98
10.29.1.2 Index(cudalImageListf &images, int x, int y, int img)	98
10.29.1.3 Interpolated(cudalImageListf &images, float x, float y, int img, bool &outside)	99
10.29.1.4 UnbindTexture(cudalImageListf &images)	99
10.30 ImageSampler_SimpleTextureRead Class Reference	99
10.30.1 Member Function Documentation	99
10.30.1.1 BindTexture(cudalImageListf &images)	99
10.30.1.2 Index(cudalImageListf &images, int x, int y, int img)	99
10.30.1.3 Interpolated(cudalImageListf &images, float x, float y, int img, bool &outside)	100
10.30.1.4 ofs(float x)	100
10.30.1.5 UnbindTexture(cudalImageListf &images)	100
10.31 QueuedCPUTracker::Job Struct Reference	100
10.31.1 Constructor & Destructor Documentation	100
10.31.1.1 Job()	100
10.31.1.2 ~Job()	101
10.31.2 Member Data Documentation	101
10.31.2.1 data	101
10.31.2.2 dataType	101
10.31.2.3 job	101
10.32 Image4DCudaArray< T >::KernelInst Struct Reference	101
10.32.1 Member Function Documentation	101
10.32.1.1 getImagePos(int image)	101
10.32.1.2 readSurfacePixel(surface< void, cudaSurfaceType2DLayered > surf, int x, int y, int z)	102

10.32.1.3 writeSurfacePixel(surface< void, cudaSurfaceType2DLayered > surf, int x, int y, int z, T value)	102
10.32.2 Member Data Documentation	102
10.32.2.1 img _h	102
10.32.2.2 img _w	102
10.32.2.3 layer _w	102
10.33 Image4DMemory< T >::KernelParams Struct Reference	102
10.33.1 Member Function Documentation	103
10.33.1.1 GetImagePos(int z, int l)	103
10.33.2 Member Data Documentation	103
10.33.2.1 cols	103
10.33.2.2 d_data	103
10.33.2.3 depth	103
10.33.2.4 img _h	103
10.33.2.5 img _w	103
10.33.2.6 pitch	103
10.34 QueuedCUDATracker::KernelProfileTime Struct Reference	103
10.34.1 Constructor & Destructor Documentation	104
10.34.1.1 KernelProfileTime()	104
10.34.2 Member Data Documentation	104
10.34.2.1 com	104
10.34.2.2 getResults	104
10.34.2.3 imageCopy	104
10.34.2.4 qi	104
10.34.2.5 zcompute	104
10.34.2.6 zlutAlign	104
10.35 kissfft< T_Scalar, T_traits > Class Template Reference	104
10.35.1 Member Typedef Documentation	105
10.35.1.1 cpx_type	105
10.35.1.2 scalar_type	105
10.35.1.3 traits_type	105

10.35.2 Constructor & Destructor Documentation	105
10.35.2.1 kissfft(int nfft, bool inverse, const traits_type &traits=traits_type())	105
10.35.3 Member Function Documentation	105
10.35.3.1 C_ADD(cpx_type &c, const cpx_type &a, const cpx_type &b)	105
10.35.3.2 C_ADDTO(cpx_type &c, const cpx_type &a)	106
10.35.3.3 C_FIXDIV(cpx_type &, int)	106
10.35.3.4 C_MUL(cpx_type &c, const cpx_type &a, const cpx_type &b)	106
10.35.3.5 C_MULBYSCALAR(cpx_type &c, const scalar_type &a)	106
10.35.3.6 C_SUB(cpx_type &c, const cpx_type &a, const cpx_type &b)	106
10.35.3.7 HALF_OF(const scalar_type &a)	106
10.35.3.8 kf_bfly2(cpx_type *Fout, const size_t fstride, int m)	106
10.35.3.9 kf_bfly3(cpx_type *Fout, const size_t fstride, const size_t m)	107
10.35.3.10kf_bfly4(cpx_type *Fout, const size_t fstride, const size_t m)	107
10.35.3.11kf_bfly5(cpx_type *Fout, const size_t fstride, const size_t m)	107
10.35.3.12kf_bfly_generic(cpx_type *Fout, const size_t fstride, int m, int p)	108
10.35.3.13kf_work(int stage, cpx_type *Fout, const cpx_type *f, size_t fstride, size_t in_stride)	109
10.35.3.14nfft()	109
10.35.3.15S_MUL(const scalar_type &a, const scalar_type &b)	109
10.35.3.16transform(const cpx_type *src, cpx_type *dst)	110
10.35.4 Member Data Documentation	110
10.35.4.1 _inverse	110
10.35.4.2 _nfft	110
10.35.4.3 _stageRadix	110
10.35.4.4 _stageRemainder	110
10.35.4.5 _twiddles	110
10.36 LocalizationJob Struct Reference	110
10.36.1 Detailed Description	111
10.36.2 Constructor & Destructor Documentation	111
10.36.2.1 LocalizationJob()	111
10.36.2.2 LocalizationJob(uint frame, uint timestamp, uint zlutPlane, uint zlutIndex)	111

10.36.3 Member Data Documentation	111
10.36.3.1 frame	111
10.36.3.2 initialPos	111
10.36.3.3 timestamp	111
10.36.3.4 zlutIndex	112
10.37 LocalizationParams Struct Reference	112
10.37.1 Member Data Documentation	112
10.37.1.1 zlutIndex	112
10.37.1.2 zlutPlane	112
10.38 LocalizationResult Struct Reference	112
10.38.1 Detailed Description	113
10.38.2 Member Function Documentation	113
10.38.2.1 pos2D()	113
10.38.3 Member Data Documentation	113
10.38.3.1 error	113
10.38.3.2 firstGuess	113
10.38.3.3 imageMean	113
10.38.3.4 job	113
10.38.3.5 pos	114
10.39 LsqSqQuadFit< T > Class Template Reference	114
10.39.1 Constructor & Destructor Documentation	114
10.39.1.1 LsqSqQuadFit(uint numPts, const T *xval, const T *yval, const T *weights=0)	114
10.39.1.2 LsqSqQuadFit()	115
10.39.2 Member Function Documentation	115
10.39.2.1 calculate(uint numPts, const T *X, const T *Y, const T *weights)	115
10.39.2.2 compute(T pos)	115
10.39.2.3 computeDeriv(T pos)	115
10.39.2.4 computeSums(const T *X, const T *Y, const T *weights, uint numPts)	115
10.39.2.5 maxPos()	116
10.39.2.6 vertexForm()	116

10.39.3 Member Data Documentation	116
10.39.3.1 a	116
10.39.3.2 b	116
10.39.3.3 c	116
10.39.3.4 d	116
10.39.3.5 h	117
10.39.3.6 k	117
10.39.3.7 xoffset	117
10.40LVArray< T > Struct Template Reference	117
10.40.1 Member Data Documentation	117
10.40.1.1 dimSize	117
10.40.1.2 elem	117
10.41LVArray2D< T > Struct Template Reference	117
10.41.1 Member Function Documentation	118
10.41.1.1 get(int row, int col)	118
10.41.1.2 numElem()	118
10.41.1.3 xy(int col, int row)	118
10.41.2 Member Data Documentation	118
10.41.2.1 dimSizes	118
10.41.2.2 elem	118
10.42LVArray3D< T > Struct Template Reference	118
10.42.1 Member Function Documentation	119
10.42.1.1 numElem()	119
10.42.2 Member Data Documentation	119
10.42.2.1 dimSizes	119
10.42.2.2 elem	119
10.43LVArrayND< T, N > Struct Template Reference	119
10.43.1 Member Function Documentation	119
10.43.1.1 numElem()	119
10.43.2 Member Data Documentation	119

10.43.2.1 dimSizes	119
10.43.2.2 elem	119
10.44LVDataType< T > Struct Template Reference	120
10.45LVDataType< double > Struct Template Reference	120
10.45.1 Member Enumeration Documentation	120
10.45.1.1 anonymous enum	120
10.46LVDataType< float > Struct Template Reference	120
10.46.1 Member Enumeration Documentation	121
10.46.1.1 anonymous enum	121
10.47LVDataType< int16_t > Struct Template Reference	121
10.47.1 Member Enumeration Documentation	121
10.47.1.1 anonymous enum	121
10.48LVDataType< int32_t > Struct Template Reference	121
10.48.1 Member Enumeration Documentation	122
10.48.1.1 anonymous enum	122
10.49LVDataType< int64_t > Struct Template Reference	122
10.49.1 Member Enumeration Documentation	122
10.49.1.1 anonymous enum	122
10.50LVDataType< int8_t > Struct Template Reference	122
10.50.1 Member Enumeration Documentation	123
10.50.1.1 anonymous enum	123
10.51LVDataType< std::complex< double > > Struct Template Reference	123
10.51.1 Member Enumeration Documentation	123
10.51.1.1 anonymous enum	123
10.52LVDataType< std::complex< float > > Struct Template Reference	123
10.52.1 Member Enumeration Documentation	124
10.52.1.1 anonymous enum	124
10.53LVDataType< uint16_t > Struct Template Reference	124
10.53.1 Member Enumeration Documentation	124
10.53.1.1 anonymous enum	124

10.54LVDataType< uint32_t > Struct Template Reference	124
10.54.1 Member Enumeration Documentation	125
10.54.1.1 anonymous enum	125
10.55LVDataType< uint64_t > Struct Template Reference	125
10.55.1 Member Enumeration Documentation	125
10.55.1.1 anonymous enum	125
10.56LVDataType< uint8_t > Struct Template Reference	125
10.56.1 Member Enumeration Documentation	126
10.56.1.1 anonymous enum	126
10.57Matrix3X3 Class Reference	126
10.57.1 Constructor & Destructor Documentation	127
10.57.1.1 Matrix3X3()	127
10.57.1.2 Matrix3X3(vector3f x, vector3f y, vector3f z)	127
10.57.2 Member Function Documentation	127
10.57.2.1 at(int i, int j)	127
10.57.2.2 at(int i, int j) const	127
10.57.2.3 dbgprint()	127
10.57.2.4 Determinant() const	127
10.57.2.5 diag() const	127
10.57.2.6 Inverse() const	128
10.57.2.7 InverseTranspose() const	128
10.57.2.8 operator()(int i, int j)	128
10.57.2.9 operator()(int i, int j) const	128
10.57.2.10 operator*=(float a)	128
10.57.2.11 operator+=(const Matrix3X3 &b)	129
10.57.2.12 operator[](int i)	129
10.57.2.13 operator[](int i) const	129
10.57.2.14 row(int i)	129
10.57.2.15 row(int i) const	129
10.57.2.16 test()	129

10.57.3 Member Data Documentation	129
10.57.3.1 m	129
10.58 MeasureTime Struct Reference	129
10.58.1 Constructor & Destructor Documentation	130
10.58.1.1 MeasureTime(const char *name)	130
10.58.1.2 ~MeasureTime()	130
10.58.2 Member Data Documentation	130
10.58.2.1 freq	130
10.58.2.2 name	130
10.58.2.3 time	130
10.59 Threads::Mutex Struct Reference	130
10.59.1 Constructor & Destructor Documentation	131
10.59.1.1 Mutex(const char *name=0)	131
10.59.1.2 ~Mutex()	131
10.59.2 Member Function Documentation	131
10.59.2.1 lock()	131
10.59.2.2 msg(const char *m)	131
10.59.2.3 unlock()	132
10.59.3 Member Data Documentation	132
10.59.3.1 h	132
10.59.3.2 lockCount	132
10.59.3.3 name	132
10.59.3.4 trace	132
10.60 my_error_mgr Struct Reference	132
10.60.1 Member Data Documentation	132
10.60.1.1 pub	132
10.61 outputter::outputModes Struct Reference	132
10.61.1 Member Data Documentation	133
10.61.1.1 Console	133
10.61.1.2 File	133

10.61.1.3 Images	133
10.62outputter Class Reference	133
10.62.1 Constructor & Destructor Documentation	134
10.62.1.1 outputter(int mode=1)	134
10.62.1.2 ~outputter()	134
10.62.2 Member Function Documentation	134
10.62.2.1 init(int mode)	134
10.62.2.2 newFile(std::string filename, const char *mode=""a"")	134
10.62.2.3 outputArray(T *arr, int size)	135
10.62.2.4 outputImage(ImageData img, std::string filename=""UsedImage"")	135
10.62.2.5 outputString(std::string out, bool ConsoleOnly=false)	135
10.62.3 Member Data Documentation	135
10.62.3.1 folder	135
10.62.3.2 modes	135
10.62.3.3 outputFile	135
10.63PathSeperator Struct Reference	135
10.63.1 Constructor & Destructor Documentation	136
10.63.1.1 PathSeperator(std::string fullpath)	136
10.63.2 Member Data Documentation	136
10.63.2.1 directory	136
10.63.2.2 extension	136
10.63.2.3 filename	136
10.64pinned_array< T, flags > Class Template Reference	136
10.64.1 Constructor & Destructor Documentation	137
10.64.1.1 pinned_array()	137
10.64.1.2 ~pinned_array()	137
10.64.1.3 pinned_array(size_t n)	137
10.64.1.4 pinned_array(const pinned_array< TOther, f > &src)	138
10.64.1.5 pinned_array(Iterator first, Iterator end)	138
10.64.1.6 pinned_array(const device_vec< T > &src)	138

10.64.2 Member Function Documentation	138
10.64.2.1 begin()	138
10.64.2.2 begin() const	138
10.64.2.3 data()	138
10.64.2.4 end()	138
10.64.2.5 end() const	138
10.64.2.6 free()	139
10.64.2.7 init(int n)	139
10.64.2.8 memsize()	139
10.64.2.9 operator=(const pinned_array< TOther, F > &src)	139
10.64.2.10 operator[](int i)	139
10.64.2.11 operator[](int i) const	139
10.64.2.12 size() const	139
10.64.3 Member Data Documentation	140
10.64.3.1 d	140
10.64.3.2 n	140
10.65 BeadFinder::Position Struct Reference	140
10.65.1 Constructor & Destructor Documentation	140
10.65.1.1 Position(int x=0, int y=0)	140
10.65.2 Member Data Documentation	140
10.65.2.1 x	140
10.65.2.2 y	140
10.66 QI Class Reference	140
10.66.1 Member Function Documentation	141
10.66.1.1 blocks(int njobs)	141
10.66.1.2 Execute(BaseKernelParams &p, const QTrkComputedConfig &cfg, Stream< Instance *s, DeviceInstance *d, device_vec< float3 > *initial, device_vec< float3 > *output)	141
10.66.1.3 Init(QTrkComputedConfig &cfg, int batchSize)	142
10.66.1.4 InitDevice(DeviceInstance *d, QTrkComputedConfig &cc)	142
10.66.1.5 InitStream(StreamInstance *s, QTrkComputedConfig &cc, cudaStream_t stream, int batchSize)	142

10.66.1.6 Iterate(BaseKernelParams &p, device_vec< float3 > *initial, device_vec< float3 > *output, StreamInstance *s, DeviceInstance *d, int angularSteps)	143
10.66.1.7 threads()	143
10.66.2 Member Data Documentation	144
10.66.2.1 batchSize	144
10.66.2.2 numThreads	144
10.66.2.3 params	144
10.66.2.4 qi_FFT_length	144
10.67 QIParams Struct Reference	144
10.67.1 Member Data Documentation	144
10.67.1.1 cos_sin_table	144
10.67.1.2 iterations	144
10.67.1.3 maxRadius	144
10.67.1.4 minRadius	144
10.67.1.5 radialSteps	144
10.67.1.6 trigtablesize	144
10.68 QTrkComputedConfig Struct Reference	145
10.68.1 Detailed Description	145
10.68.2 Constructor & Destructor Documentation	146
10.68.2.1 QTrkComputedConfig()	146
10.68.2.2 QTrkComputedConfig(const QTrkSettings &base)	146
10.68.3 Member Function Documentation	146
10.68.3.1 Update()	146
10.68.3.2 WriteToFile()	146
10.68.3.3 WriteToLog()	147
10.68.4 Member Data Documentation	147
10.68.4.1 qi_angstepspq	147
10.68.4.2 qi_maxradius	147
10.68.4.3 qi_radialsteps	147
10.68.4.4 zlut-angularsteps	147
10.68.4.5 zlut_maxradius	147

10.68.4.6 zlut_radialsteps	148
10.69 QTrkSettings Struct Reference	148
10.69.1 Detailed Description	149
10.69.2 Constructor & Destructor Documentation	150
10.69.2.1 QTrkSettings()	150
10.69.3 Member Data Documentation	150
10.69.3.1 com_bgcorrection	150
10.69.3.2 cuda_device	150
10.69.3.3 downsample	150
10.69.3.4 gauss2D_iterations	150
10.69.3.5 gauss2D_sigma	151
10.69.3.6 height	151
10.69.3.7 numThreads	151
10.69.3.8 qi_angstep_factor	151
10.69.3.9 qi_angular_coverage	151
10.69.3.10qi_iterations	151
10.69.3.11qi_minradius	151
10.69.3.12qi_radial_coverage	151
10.69.3.13qi_roi_coverage	151
10.69.3.14testRun	152
10.69.3.15width	152
10.69.3.16xc1_iterations	152
10.69.3.17xc1_profileLength	152
10.69.3.18xc1_profileWidth	152
10.69.3.19zlut-angular_coverage	152
10.69.3.20zlut_minradius	152
10.69.3.21zlut_radial_coverage	152
10.69.3.22zlut_roi_coverage	153
10.70 QueuedCPUTracker Class Reference	153
10.70.1 Detailed Description	155

10.70.2 Constructor & Destructor Documentation	155
10.70.2.1 QueuedCPUTracker(const QTrkComputedConfig &cc)	155
10.70.2.2 ~QueuedCPUTracker()	156
10.70.3 Member Function Documentation	156
10.70.3.1 AddJob(Job *)	156
10.70.3.2 AllocateJob()	156
10.70.3.3 ApplyOffsetGain(CPUTracker *trk, int beadIndex)	157
10.70.3.4 BeginLUT(uint flags)	157
10.70.3.5 Break(bool pause)	157
10.70.3.6 BuildLUT(void *data, int pitch, QTRK_PixelDataType pdt, int plane, vector2f *known_pos=0) override	157
10.70.3.7 ClearResults() override	158
10.70.3.8 EnableRadialZLUTCompareProfile(bool enabled)	158
10.70.3.9 FetchResults(LocalizationResult *results, int maxResults) override	159
10.70.3.10 FinalizeLUT() override	159
10.70.3.11 Flush() override	160
10.70.3.12 GenerateTestImage(float *dst, float xp, float yp, float z, float photoncount)	160
10.70.3.13 GetConfigValues() override	160
10.70.3.14 GetDebugImage(int id, int *w, int *h, float **data)	160
10.70.3.15 GetImageLUTByIndex(int index, int plane=0)	161
10.70.3.16 GetImageZLUT(float *dst)	161
10.70.3.17 GetImageZLUTSize(int *dims)	161
10.70.3.18 GetNextJob()	161
10.70.3.19 GetProfileReport()	161
10.70.3.20 GetQueueLength(int *maxQueueLength=0) override	162
10.70.3.21 GetRadialZLUT(float *zlut) override	162
10.70.3.22 GetRadialZLUTCompareProfile(float *dst)	162
10.70.3.23 GetRadialZLUTSize(int &count, int &planes, int &rsteps) override	162
10.70.3.24 GetResultCount() override	163
10.70.3.25 GetZLUTByIndex(int index)	163
10.70.3.26 ImageLUTHeight()	163

10.70.3.27	imageLUTNumBeads()	163
10.70.3.28	imageLUTWidth()	163
10.70.3.29	sIdle() override	163
10.70.3.30	JobFinished(Job *j)	163
10.70.3.31	NumThreads()	164
10.70.3.32	ProcessJob(Thread *th, Job *j)	164
10.70.3.33	ScheduleLocalization(void *data, int pitch, QTRK_PixelDataType pdt, const LocalizationJob *jobInfo) override	165
10.70.3.34	SetConfigValue(std::string name, std::string value) override	166
10.70.3.35	SetImageZLUT(float *dst, float *radial_zlut, int *dims)	166
10.70.3.36	SetLocalizationMode(LocMode_t lt) override	166
10.70.3.37	SetPixelCalibrationFactors(float offsetFactor, float gainFactor) override	167
10.70.3.38	SetPixelCalibrationImages(float *offset, float *gain) override	167
10.70.3.39	SetRadialWeights(float *rweights) override	167
10.70.3.40	SetRadialWeights(std::vector< float > weights)	168
10.70.3.41	SetRadialZLUT(float *data, int num_zluts, int planes) override	168
10.70.3.42	SetTrackerImage(CPUTracker *trk, Job *job)	168
10.70.3.43	Start()	168
10.70.3.44	UpdateZLUTs()	169
10.70.3.45	WorkerThreadMain(void *arg)	169
10.70.3.46	ZLUTSelfTest()	169
10.70.4	Member Data Documentation	170
10.70.4.1	calib_gain	170
10.70.4.2	calib_offset	170
10.70.4.3	dbgPrintResults	170
10.70.4.4	downsampleHeight	170
10.70.4.5	downsampleWidth	170
10.70.4.6	gc_gainFactor	170
10.70.4.7	gc_mutex	170
10.70.4.8	gc_offsetFactor	170
10.70.4.9	image_lut	170

10.70.4.10	image_lut_dims	170
10.70.4.11	image_lut_dz	170
10.70.4.12	image_lut_dz2	170
10.70.4.13	image_lut_nElem_per_bead	170
10.70.4.14	jobCount	170
10.70.4.15	jobs	170
10.70.4.16	jobs_buffer	170
10.70.4.17	jobs_buffer_mutex	170
10.70.4.18	jobs_mutex	170
10.70.4.19	jobsInProgress	170
10.70.4.20	localizeMode	170
10.70.4.21	maxQueueSize	170
10.70.4.22	processJobs	170
10.70.4.23	qi_radialbinweights	171
10.70.4.24	quitWork	171
10.70.4.25	resultCode	171
10.70.4.26	results	171
10.70.4.27	results_mutex	171
10.70.4.28	threads	171
10.70.4.29	zcmp	171
10.70.4.30	zlut_buildflags	171
10.70.4.31	zlut_cmpprofiles	171
10.70.4.32	zlut_count	171
10.70.4.33	zlut_enablecmpprof	171
10.70.4.34	zlut_planes	171
10.70.4.35	zluts	171
10.71	QueuedCUDATracker Class Reference	171
10.71.1	Detailed Description	173
10.71.2	Member Typedef Documentation	174
10.71.2.1	ImageLUT	174

10.71.3 Constructor & Destructor Documentation	174
10.71.3.1 QueuedCUDATracker(const QTrkComputedConfig &cc, int batchSize=-1)	174
10.71.3.2 ~QueuedCUDATracker()	175
10.71.4 Member Function Documentation	175
10.71.4.1 BeginLUT(uint flags) override	175
10.71.4.2 blocks(int workItems)	175
10.71.4.3 blocks()	175
10.71.4.4 BuildLUT(void *data, int pitch, QTRK_PixelDataType pdt, int plane, vector2f *known_pos=0) override	176
10.71.4.5 ClearResults() override	176
10.71.4.6 CopyStreamResults(Stream *s)	177
10.71.4.7 CPU_ApplyOffsetGain(CPUTracker *trk, int beadIndex)	177
10.71.4.8 CreateStream(Device *device, int streamIndex)	177
10.71.4.9 EnableRadialZLUTCompareProfile(bool enabled)	178
10.71.4.10 EnableTextureCache(bool useTextureCache)	178
10.71.4.11 ExecuteBatch(Stream *s)	178
10.71.4.12 FetchResults(LocalizationResult *results, int maxResults) override	180
10.71.4.13 FinalizeLUT() override	180
10.71.4.14 Flush() override	180
10.71.4.15 GetConfigValues() override	181
10.71.4.16 GetProfileReport() override	181
10.71.4.17 GetQueueLength(int *maxQueueLen) override	181
10.71.4.18 GetRadialZLUT(float *data) override	181
10.71.4.19 GetRadialZLUTCompareProfile(float *dst)	182
10.71.4.20 GetRadialZLUTSize(int &count, int &planes, int &radialSteps) override	182
10.71.4.21 GetReadyStream()	182
10.71.4.22 GetResultCount() override	182
10.71.4.23 InitializeDeviceList()	183
10.71.4.24 IsIdle() override	183
10.71.4.25 ScheduleLocalization(void *data, int pitch, QTRK_PixelDataType pdt, const LocalizationJob *jobInfo) override	183

10.71.4.26SchedulingThreadEntryPoint(void *param)	184
10.71.4.27SchedulingThreadMain()	184
10.71.4.28SetConfigValue(std::string name, std::string value) override	185
10.71.4.29SetLocalizationMode(LocMode_t locType) override	185
10.71.4.30SetPixelCalibrationFactors(float offsetFactor, float gainFactor) override	185
10.71.4.31SetPixelCalibrationImages(float *offset, float *gain) override	185
10.71.4.32SetRadialWeights(float *zcmp) override	186
10.71.4.33SetRadialZLUT(float *data, int numLUTs, int planes) override	186
10.71.4.34StreamUpdateZLUTSize(Stream *s)	186
10.71.4.35threads()	186
10.71.5 Member Data Documentation	187
10.71.5.1 batchesDone	187
10.71.5.2 batchSize	187
10.71.5.3 cpu_time	187
10.71.5.4 deviceProp	187
10.71.5.5 deviceReport	187
10.71.5.6 devices	187
10.71.5.7 gc_gain	187
10.71.5.8 gc_gainFactor	187
10.71.5.9 gc_mutex	187
10.71.5.10gc_offset	187
10.71.5.11gc_offsetFactor	187
10.71.5.12jobQueueMutex	187
10.71.5.13localizeMode	187
10.71.5.14numThreads	187
10.71.5.15qalign	187
10.71.5.16qi	187
10.71.5.17quitScheduler	187
10.71.5.18resultCode	187
10.71.5.19resultMutex	188

10.71.5.20results	188
10.71.5.21schedulingThread	188
10.71.5.22streams	188
10.71.5.23time	188
10.71.5.24useTextureCache	188
10.71.5.25zlut_build_flags	188
10.72QueuedTracker Class Reference	188
10.72.1 Detailed Description	190
10.72.2 Member Typedef Documentation	190
10.72.2.1 ConfigValueMap	190
10.72.3 Constructor & Destructor Documentation	190
10.72.3.1 QueuedTracker()	190
10.72.3.2 ~QueuedTracker()	190
10.72.4 Member Function Documentation	190
10.72.4.1 BeginLUT(uint flags)=0	190
10.72.4.2 BuildLUT(void *data, int pitch, QTRK_PixelDataType pdt, int plane, vector2f *known_pos=0)	190
10.72.4.3 ClearResults()=0	190
10.72.4.4 ComputeZBiasCorrection(int bias_planes, ClImageData *result, int smpPerPixel, bool useSplineInterp)	191
10.72.4.5 DebugImage(int ID)	191
10.72.4.6 EnableRadialZLUTCompareProfile(bool enabled)=0	191
10.72.4.7 FetchResults(LocalizationResult *results, int maxResults)=0	192
10.72.4.8 FinalizeLUT()=0	192
10.72.4.9 Flush()=0	192
10.72.4.10GetConfigValues()=0	192
10.72.4.11GetDebugImage(int ID, int *w, int *h, float **pData)	192
10.72.4.12GetImageZLUT(float *dst)	192
10.72.4.13GetImageZLUTSize(int *dims)	192
10.72.4.14GetProfileReport()	192
10.72.4.15GetQueueLength(int *maxQueueLen=0)=0	193

10.72.4.16GetRadialZLUT(float *dst)=0	193
10.72.4.17GetRadialZLUTCompareProfile(float *dst)=0	193
10.72.4.18GetRadialZLUTSize(int &count, int &planes, int &radialsteps)=0	193
10.72.4.19GetResultCount()=0	193
10.72.4.20GetWarnings()	193
10.72.4.21GetZLUTBiasCorrection()	193
10.72.4.22IsIdle()=0	193
10.72.4.23ScheduleFrame(void *imgptr, int pitch, int width, int height, ROIPosition *positions, int numROI, QTRK_PixelDataType pdt, const LocalizationJob *jobInfo)	194
10.72.4.24ScheduleImageData(ImageData *data, const LocalizationJob *jobInfo)	194
10.72.4.25ScheduleLocalization(void *data, int pitch, QTRK_PixelDataType pdt, const LocalizationJob *jobInfo)=0	194
10.72.4.26ScheduleLocalization(uchar *data, int pitch, QTRK_PixelDataType pdt, uint frame, uint timestamp, vector3f *initial, uint zlutIndex)	194
10.72.4.27SetConfigValue(std::string name, std::string value)=0	195
10.72.4.28SetImageZLUT(float *dst, float *radial_zlut, int *dims)	195
10.72.4.29SetLocalizationMode(LocMode_t locType)=0	195
10.72.4.30SetPixelCalibrationFactors(float offsetFactor, float gainFactor)=0	195
10.72.4.31SetPixelCalibrationImages(float *offset, float *gain)=0	195
10.72.4.32SetRadialWeights(float *zcmp)=0	195
10.72.4.33SetRadialZLUT(float *data, int count, int planes)=0	195
10.72.4.34SetZLUTBiasCorrection(const CImageData &data)	195
10.72.4.35ZLUTBiasCorrection(float z, int zlut_planes, int bead)	196
10.72.5 Member Data Documentation	196
10.72.5.1 cfg	196
10.72.5.2 zlut_bias_correction	196
10.73 ResultFile Class Reference	196
10.73.1 Constructor & Destructor Documentation	197
10.73.1.1 ResultFile()	197
10.73.1.2 ~ResultFile()	197
10.73.2 Member Function Documentation	197
10.73.2.1 LoadRow(std::vector<vector3f> &pos)=0	197

10.73.2.2 SaveRow(std::vector< vector3f > &pos)=0	197
10.74 ResultManager Class Reference	197
10.74.1 Constructor & Destructor Documentation	198
10.74.1.1 ResultManager(const char *outfile, const char *frameinfo, ResultManagerConfig *cfg, std::vector< std::string > colnames)	198
10.74.1.2 ~ResultManager()	199
10.74.2 Member Function Documentation	199
10.74.2.1 CheckResultSpace(int fr)	199
10.74.2.2 Config()	199
10.74.2.3 Flush()	199
10.74.2.4 GetBeadPositions(int startFrame, int endFrame, int bead, LocalizationResult *r)	200
10.74.2.5 GetFrameCount()	200
10.74.2.6 GetFrameCounters()	200
10.74.2.7 GetResults(LocalizationResult *results, int startFrame, int numResults)	200
10.74.2.8 GetTracker()	200
10.74.2.9 RemoveBeadResults(int bead)	201
10.74.2.10 SaveSection(int start, int end, const char *beadposfile, const char *infofile)	201
10.74.2.11 SetTracker(QueuedTracker *qtrk)	201
10.74.2.12 StoreFrameInfo(int frame, double timestamp, float *columns)	201
10.74.2.13 StoreResult(LocalizationResult *r)	201
10.74.2.14 ThreadLoop(void *param)	202
10.74.2.15 Update()	202
10.74.2.16 Write()	203
10.74.2.17 WriteBinaryFileHeader()	203
10.74.2.18 WriteBinaryResults()	203
10.74.2.19 WriteTextResults()	204
10.74.3 Member Data Documentation	204
10.74.3.1 cnt	204
10.74.3.2 config	204
10.74.3.3 frameInfoFile	204
10.74.3.4 frameInfoNames	204

10.74.3.5 frameResults	204
10.74.3.6 outputFile	204
10.74.3.7 qtrk	204
10.74.3.8 quit	204
10.74.3.9 resultFile	204
10.74.3.10resultMutex	204
10.74.3.11thread	204
10.74.3.12trackerMutex	204
10.75ResultManagerConfig Struct Reference	205
10.75.1 Member Data Documentation	205
10.75.1.1 binaryOutput	205
10.75.1.2 maxFramesInMemory	205
10.75.1.3 numBeads	205
10.75.1.4 numFrameInfoColumns	205
10.75.1.5 offset	205
10.75.1.6 scaling	205
10.75.1.7 writeInterval	205
10.76ROIPosition Struct Reference	205
10.76.1 Detailed Description	206
10.76.2 Member Data Documentation	206
10.76.2.1 x	206
10.76.2.2 y	206
10.77RunTrackerResults Struct Reference	206
10.77.1 Member Function Documentation	206
10.77.1.1 computeStats()	206
10.77.2 Member Data Documentation	207
10.77.2.1 meanErr	207
10.77.2.2 output	207
10.77.2.3 stdev	207
10.77.2.4 truepos	207

10.78 SampleFisherMatrix Class Reference	207
10.78.1 Constructor & Destructor Documentation	207
10.78.1.1 SampleFisherMatrix(float maxValue)	207
10.78.2 Member Function Documentation	208
10.78.2.1 Compute(vector3f pos, vector3f delta, ImageData &lut, int w, int h, float zlutMinRadius, float zlutMaxRadius)	208
10.78.2.2 Compute(vector3f pos, vector3f delta, int w, int h, std::function< void(ImageData &out, vector3f pos)> imageGenerator)	208
10.78.2.3 ComputeAverageFisher(vector3f pos, int Nsamples, vector3f sampleRange, vector3f delta, int w, int h, std::function< void(ImageData &out, vector3f pos)> imggen)	209
10.78.2.4 FisherElem(TImageData< T > &mu, TImageData< T > &muderiv1, TImageData< T > &muderiv2)	209
10.78.2.5 ImgDeriv(TImageData< T > &imgpx, TImageData< T > &imgmx, T delta)	209
10.78.3 Member Data Documentation	209
10.78.3.1 maxValue	209
10.79 ScopedCPUProfiler Class Reference	210
10.79.1 Constructor & Destructor Documentation	210
10.79.1.1 ScopedCPUProfiler(double *time)	210
10.79.1.2 ~ScopedCPUProfiler()	210
10.79.2 Member Data Documentation	210
10.79.2.1 start	210
10.79.2.2 time	210
10.80 SpeedAccResult Struct Reference	210
10.80.1 Member Function Documentation	211
10.80.1.1 Compute(const std::vector< vector3f > &results, std::function< vector3f(int x) > truepos)	211
10.80.2 Member Data Documentation	211
10.80.2.1 acc	211
10.80.2.2 bias	211
10.80.2.3 crlb	211
10.80.2.4 speed	211
10.81 SpeedInfo Struct Reference	211
10.81.1 Member Data Documentation	212

10.81.1.1 <code>sched_cpu</code>	212
10.81.1.2 <code>sched_gpu</code>	212
10.81.1.3 <code>speed_cpu</code>	212
10.81.1.4 <code>speed_gpu</code>	212
10.82 <code>QueuedCUDATracker::Stream</code> Struct Reference	212
10.82.1 Member Enumeration Documentation	213
10.82.1.1 <code>State</code>	213
10.82.2 Constructor & Destructor Documentation	213
10.82.2.1 <code>Stream(int streamIndex)</code>	213
10.82.2.2 <code>~Stream()</code>	213
10.82.3 Member Function Documentation	214
10.82.3.1 <code>IsExecutionDone()</code>	214
10.82.3.2 <code>JobCount()</code>	214
10.82.3.3 <code>OutputMemoryUse()</code>	214
10.82.4 Member Data Documentation	214
10.82.4.1 <code>batchStart</code>	214
10.82.4.2 <code>com</code>	214
10.82.4.3 <code>comDone</code>	214
10.82.4.4 <code>d_com</code>	214
10.82.4.5 <code>d_imgmeans</code>	214
10.82.4.6 <code>d_locParams</code>	214
10.82.4.7 <code>d_radialprofiles</code>	214
10.82.4.8 <code>d_resultpos</code>	214
10.82.4.9 <code>d_zlutcmpscores</code>	214
10.82.4.10 <code>device</code>	214
10.82.4.11 <code>hostImageBuf</code>	215
10.82.4.12 <code>imageBufMutex</code>	215
10.82.4.13 <code>imageCopyDone</code>	215
10.82.4.14 <code>images</code>	215
10.82.4.15 <code>mapDone</code>	215

10.82.4.16mgMeans	215
10.82.4.17jobs	215
10.82.4.18ocalizationDone	215
10.82.4.19ocalizeFlags	215
10.82.4.20ocParams	215
10.82.4.21qalign_instance	215
10.82.4.22qalignDone	215
10.82.4.23qi_instance	215
10.82.4.24qiDone	215
10.82.4.25results	215
10.82.4.26state	215
10.82.4.27stream	215
10.82.4.28zcomputeDone	215
10.83QL::StreamInstance Struct Reference	215
10.83.1 Constructor & Destructor Documentation	216
10.83.1.1 ~StreamInstance()	216
10.83.2 Member Function Documentation	216
10.83.2.1 memsize()	216
10.83.3 Member Data Documentation	216
10.83.3.1 d_Qlprofiles	216
10.83.3.2 d_Qlprofiles_reverse	216
10.83.3.3 d_quadrants	216
10.83.3.4 d_shiftbuffer	216
10.83.3.5 fftPlan	216
10.83.3.6 stream	216
10.84TextResultFile Class Reference	217
10.84.1 Constructor & Destructor Documentation	217
10.84.1.1 TextResultFile(const char *fn, bool write)	217
10.84.2 Member Function Documentation	217
10.84.2.1 LoadRow(std::vector< vector3f > &pos)	217

10.84.2.2 SaveRow(std::vector< vector3f > &pos)	217
10.84.3 Member Data Documentation	218
10.84.3.1 f	218
10.85QueuedCPUTracker::Thread Struct Reference	218
10.85.1 Constructor & Destructor Documentation	218
10.85.1.1 Thread()	218
10.85.2 Member Function Documentation	218
10.85.2.1 lock()	218
10.85.2.2 unlock()	218
10.85.3 Member Data Documentation	219
10.85.3.1 manager	219
10.85.3.2 mutex	219
10.85.3.3 thread	219
10.85.3.4 tracker	219
10.86 ThreadPool< TWorkItem, TFunctor > Class Template Reference	219
10.86.1 Constructor & Destructor Documentation	220
10.86.1.1 ThreadPool(TFunctor f, int Nthreads=-1)	220
10.86.1.2 ~ThreadPool()	220
10.86.2 Member Function Documentation	220
10.86.2.1 AddWork(TWorkItem w)	220
10.86.2.2 GetNewItem(TWorkItem &item)	220
10.86.2.3 IsDone()	220
10.86.2.4 ItemDone()	221
10.86.2.5 ProcessArray(TWorkItem *items, int n)	221
10.86.2.6 Quit()	221
10.86.2.7 ThreadEntryPoint(void *param)	221
10.86.2.8 WaitUntilDone()	221
10.86.3 Member Data Documentation	222
10.86.3.1 inProgress	222
10.86.3.2 quit	222

10.86.3.3 threads	222
10.86.3.4 work	222
10.86.3.5 worker	222
10.86.3.6 workMutex	222
10.87 Threads Struct Reference	222
10.87.1 Member Typedef Documentation	223
10.87.1.1 ThreadEntryPoint	223
10.87.2 Member Function Documentation	223
10.87.2.1 Create(ThreadEntryPoint method, void *param)	223
10.87.2.2 GetCPUCount()	223
10.87.2.3 RunningVistaOrBetter()	223
10.87.2.4 SetBackgroundPriority(Handle *thread, bool bg)	223
10.87.2.5 Sleep(int ms)	223
10.87.2.6 ThreadCaller(void *param)	224
10.87.2.7 WaitAndClose(Handle *h)	224
10.88 TImageData< T > Struct Template Reference	224
10.88.1 Constructor & Destructor Documentation	225
10.88.1.1 TImageData()	225
10.88.1.2 TImageData(T *d, int w, int h)	225
10.88.2 Member Function Documentation	225
10.88.2.1 alloc(int w, int h)	225
10.88.2.2 at(int x, int y)	225
10.88.2.3 copyTo(float *dst)	225
10.88.2.4 free()	225
10.88.2.5 interpolate(float x, float y, bool *outside=0)	225
10.88.2.6 interpolate1D(int y, float x)	226
10.88.2.7 mean()	226
10.88.2.8 normalize()	226
10.88.2.9 numPixels() const	226
10.88.2.10 operator[](int i)	226

10.88.2.11pitch() const	226
10.88.2.12set(Ta *src)	226
10.88.2.13set(const TImageData< Ta > &src)	226
10.88.2.14writeAsCSV(const char *filename, const char *labels[] = 0)	227
10.88.3 Member Data Documentation	227
10.88.3.1 data	227
10.88.3.2 h	227
10.88.3.3 w	227
10.89kissfft_utils::traits< T_scalar > Struct Template Reference	227
10.89.1 Member Typedef Documentation	227
10.89.1.1 cpx_type	227
10.89.1.2 scalar_type	227
10.89.2 Member Function Documentation	227
10.89.2.1 fill_twiddles(std::complex< T_scalar > *dst, int nfft, bool inverse) const	227
10.89.2.2 prepare(std::vector< std::complex< T_scalar > > &dst, int nfft, bool inverse, std::vector< int > &stageRadix, std::vector< int > &stageRemainder) const	228
10.90vector2< T > Struct Template Reference	228
10.90.1 Constructor & Destructor Documentation	228
10.90.1.1 vector2()	228
10.90.1.2 vector2(Tx X, Ty Y)	229
10.90.2 Member Function Documentation	229
10.90.2.1 random(vector2 center, T R)	229
10.90.3 Member Data Documentation	229
10.90.3.1 x	229
10.90.3.2 y	229
10.91vector3< T > Struct Template Reference	229
10.91.1 Constructor & Destructor Documentation	230
10.91.1.1 vector3()	230
10.91.1.2 vector3(Tx X, Ty Y, Tz Z)	230
10.91.1.3 vector3(const vector3< Tc > &o)	230
10.91.2 Member Function Documentation	230

10.91.2.1 length()	230
10.91.2.2 operator*(const vector3 &o) const	230
10.91.2.3 operator*(T a) const	231
10.91.2.4 operator*=(const vector3 &o)	231
10.91.2.5 operator*=(T a)	231
10.91.2.6 operator+(const vector3 &o) const	231
10.91.2.7 operator+(float a) const	231
10.91.2.8 operator+=(const vector3 &o)	231
10.91.2.9 operator-(const vector3 &o) const	231
10.91.2.10 operator-(float a) const	231
10.91.2.11 operator-() const	232
10.91.2.12 operator-=(const vector3 &o)	232
10.91.2.13 operator/(T a)	232
10.91.2.14 operator/=(T a)	232
10.91.2.15 xy()	232
10.91.3 Friends And Related Function Documentation	232
10.91.3.1 operator*	232
10.91.3.2 operator/	232
10.91.4 Member Data Documentation	233
10.91.4.1 x	233
10.91.4.2 y	233
10.91.4.3 z	233
10.92 XCor1DBuffer Class Reference	233
10.92.1 Constructor & Destructor Documentation	233
10.92.1.1 XCor1DBuffer(int xcorw)	233
10.92.2 Member Function Documentation	233
10.92.2.1 XCorFFTHelper(complex_t *xc, complex_t *xcr, scalar_t *result)	233
10.92.3 Member Data Documentation	234
10.92.3.1 fft_backward	234
10.92.3.2 fft_forward	234
10.92.3.3 xcorw	234
10.93 ZLUTParams Struct Reference	234
10.93.1 Member Function Documentation	234
10.93.1.1 GetRadialZLUT(int bead, int plane)	234
10.93.1.2 radialSteps()	234
10.93.2 Member Data Documentation	235
10.93.2.1 angularSteps	235
10.93.2.2 img	235
10.93.2.3 maxRadius	235
10.93.2.4 minRadius	235
10.93.2.5 planes	235
10.93.2.6 trigtable	235
10.93.2.7 zcmpwindow	235

11 File Documentation	237
11.1 cputrack-test/Benchmark.cpp File Reference	237
11.1.1 Function Documentation	238
11.1.1.1 BenchmarkConfigParamRange(int n, T QTrkSettings::*param, QTrkSettings *config, std::vector< T > param_values, const char *name, int MaxPixelValue, vector3f range)	238
11.1.1.2 BenchmarkParams()	238
11.1.1.3 BenchmarkROISizes(const char *name, int n, int MaxPixelValue, int qi_iterations, int extraFlags, float range_in_nm, float pixel_size, float lutstep, int buildLUTFlags)	239
11.1.1.4 BenchmarkZAccuracy(const char *name, int n, int MaxPixelValue)	239
11.1.2 Variable Documentation	240
11.1.2.1 ElectronsPerBit	240
11.1.2.2 img_mean	240
11.1.2.3 img_sigma	240
11.2 cputrack-test/main.cpp File Reference	240
11.2.1 Enumeration Type Documentation	242
11.2.1.1 RWeightMode	242
11.2.1.2 Tests	242
11.2.2 Function Documentation	242
11.2.2.1 AccBiasTest(ImageData &lut, QueuedTracker *trk, int N, vector3f centerpos, vector3f range, const char *name, int MaxPixelValue, int extraFlags=0)	242
11.2.2.2 AutoBeadFindTest()	243
11.2.2.3 BenchmarkParams()	243
11.2.2.4 BuildConvergenceMap(int iterations)	244
11.2.2.5 BuildZLUT(std::string folder, outputter *output)	245
11.2.2.6 ComputeRadialWeights(int rsteps, float minRadius, float maxRadius)	246
11.2.2.7 CorrectedRadialProfileTest()	246
11.2.2.8 CRP_TestGeneratedData()	247
11.2.2.9 GenerateZLUTFittingCurve(const char *lutfile)	247
11.2.2.10 main()	247
11.2.2.11 ManTest()	248
11.2.2.12 OnePixelTest()	250

11.2.2.13 OutputProfileImg()	250
11.2.2.14 PixelationErrorTest()	251
11.2.2.15 PrintMenu(outputter *output)	251
11.2.2.16 RescaleLUT(CPUTracker *trk, ImageData *lut)	251
11.2.2.17 RunCOMAndQI(ImageData img, outputter *output)	252
11.2.2.18 RunTest(Tests test, const char *image, outputter *output, int ROISize)	252
11.2.2.19 RunZTrace(std::string imagePath, std::string zlutPath, outputter *output)	252
11.2.2.20 ScatterBiasArea(int roi, float scan_width, int steps, int samples, int qi_it, float ang-step)	255
11.2.2.21 SelectTests(const char *image, int OutputMode)	256
11.2.2.22 SimpleTest()	256
11.2.2.23 SkewParam(ImageData img)	257
11.2.2.24 SmallImageTest()	257
11.2.2.25 SpeedTest()	258
11.2.2.26 TestBackground(std::vector< BeadPos > beads, ImageData orilmg, outputter *output, int ROISize)	259
11.2.2.27 TestBias()	259
11.2.2.28 TestBoundCheck()	260
11.2.2.29 TestBSplineMax(float maxpos)	260
11.2.2.30 TestFourierLUT()	260
11.2.2.31 TestFourierLUTOndataset()	261
11.2.2.32 TestInterference(std::vector< BeadPos > beads, ImageData orilmg, outputter *output, int ROISize, vector2f displacement=vector2f(60, 0))	261
11.2.2.33 TestQuadrantAlign()	261
11.2.2.34 TestROIDisplacement(std::vector< BeadPos > beads, ImageData orilmg, outputter *output, int ROISize, int maxdisplacement=0)	262
11.2.2.35 TestSkew(std::vector< BeadPos > beads, ImageData orilmg, outputter *output, int ROISize)	262
11.2.2.36 TestZLUTAlign()	262
11.2.2.37 TestZRange(const char *name, const char *lutfile, int extraFlags, int clean_lut, RWeightMode weightMode=RWNone, bool biasMap=false, bool biasCorrect=false)	263
11.2.2.38 TestZRangeBias(const char *name, const char *lutfile, bool normProf)	265
11.2.2.39 WriteRadialProf(const char *file, ImageData &d)	265

11.2.3 Variable Documentation	265
11.2.3.1 ANGSTEPF	265
11.2.3.2 InDebugMode	265
11.3 cputrack-test/SharedTests.h File Reference	266
11.3.1 Function Documentation	266
11.3.1.1 EnableGainCorrection(QueuedTracker *qtrk)	266
11.3.1.2 FetchResults(QueuedTracker *trk)	267
11.3.1.3 Gauss2DTest(int NumImages=10, int JobsPerImg=1000, bool useGC=false)	267
11.3.1.4 linspace(T a, T b, int N)	268
11.3.1.5 logspace(T a, T b, int N)	268
11.3.1.6 MeanStDevError(const std::vector< vector3f > &truepos, const std::vector< vector3f > &v, vector3f &meanErr, vector3f &stdev)	268
11.3.1.7 RandomFill(float *d, int size, float mean, float std_deviation)	269
11.3.1.8 ResampleLUT(T *qtrk, ImageData *lut, int zplanes, ImageData *newlut, const char *jpgfile=0, uint buildLUTFlags=0)	269
11.3.1.9 ResizeLUT(ImageData &lut, ImageData &resized, QTrkSettings *cfg)	269
11.3.1.10 RunTracker(const char *lutfile, QTrkSettings *cfg, bool useGC, const char *name, LocMode_t locMode, int N=2000, float noiseFactor=28, float zpos=10, ImageData *pRescaledLUT=0, vector3f samplePosRange=vector3f(1, 1, 1))	270
11.3.1.11 SpeedAccTest(ImageData &lut, QTrkSettings *cfg, int N, vector3f centerpos, vector3f range, const char *name, int MaxPixelValue, int extraFlags=0, int buildLUFlags=0)	270
11.3.1.12 TestCMOSNoiseInfluence(const char *lutfile)	271
11.3.1.13 WaitForFinish(QueuedTracker *qtrk, int N)	273
11.4 cputrack-test/testutils.cpp File Reference	273
11.4.1 Function Documentation	274
11.4.1.1 AddImages(ImageData img1, ImageData img2, vector2f displacement)	274
11.4.1.2 BackgroundMedian(ImageData img)	274
11.4.1.3 BackgroundRMS(ImageData img)	274
11.4.1.4 BackgroundStdDev(ImageData img)	274
11.4.1.5 CropImage(ImageData img, int x, int y, int w, int h)	275
11.4.1.6 DirExists(const std::string &dirName_in)	275
11.4.1.7 distance(vector2f a, vector2f b)	275

11.4.1.8	GaussMask(ImageData img, float sigma)	275
11.4.1.9	GetOuterEdges(float *out, int size, ImageData img)	275
11.4.1.10	NumFilesInDir(const std::string &dirName_in)	276
11.4.1.11	NumJpgInDir(const std::string &dirName_in)	276
11.4.1.12	ResizeImage(ImageData img, int factor)	276
11.4.1.13	SkewImage(ImageData img, float fact)	276
11.5	cputrack-test/testutils.h File Reference	277
11.5.1	Enumeration Type Documentation	277
11.5.1.1	OutputModes	277
11.5.2	Function Documentation	278
11.5.2.1	AddImages(ImageData img1, ImageData img2, vector2f displacement)	278
11.5.2.2	BackgroundMedian(ImageData img)	278
11.5.2.3	BackgroundRMS(ImageData img)	278
11.5.2.4	BackgroundStdDev(ImageData img)	278
11.5.2.5	CropImage(ImageData img, int x, int y, int w, int h)	279
11.5.2.6	DirExists(const std::string &dirName_in)	279
11.5.2.7	distance(T x, T y)	279
11.5.2.8	distance(vector2f a, vector2f b)	279
11.5.2.9	GaussMask(ImageData img, float sigma)	279
11.5.2.10	GetOuterEdges(float *out, int size, ImageData img)	280
11.5.2.11	NumFilesInDir(const std::string &dirName_in)	280
11.5.2.12	NumJpgInDir(const std::string &dirName_in)	280
11.5.2.13	ResizeImage(ImageData img, int factor)	281
11.5.2.14	SkewImage(ImageData img, float fact)	281
11.5.2.15	sq(T x)	281
11.6	cputrack/BeadFinder.cpp File Reference	281
11.6.1	Function Documentation	282
11.6.1.1	abs(std::complex< float > x)	282
11.6.1.2	ComplexToJPEGFile(const char *name, std::complex< float > *d, int w, int h, bool logMode=false)	282
11.6.1.3	FFT2D(std::complex< float > *d, int w, int h, bool inverse)	282

11.6.1.4	NextPowerOf2(int x)	282
11.6.1.5	RecenterAndFilter(ImageData *img, std::vector< Position > &pts, Config *cfg)	283
11.6.1.6	SearchArea(std::complex< float > *cimg, int w, int h, int px, int py, int dist)	283
11.7	cputrack/BeadFinder.h File Reference	284
11.8	cputrack/BenchmarkLUT.cpp File Reference	284
11.9	cputrack/BenchmarkLUT.h File Reference	284
11.10	cputrack/cpu_tracker.cpp File Reference	285
11.10.1	Macro Definition Documentation	285
11.10.1.1	MARKPIXEL	285
11.10.1.2	MARKPIXELI	285
11.10.1.3	SFFT_BOTH	285
11.10.1.4	ZLUT_CMPDATA	285
11.10.2	Function Documentation	285
11.10.2.1	clamp(int v, int a, int b)	285
11.10.2.2	conjugate(const T &v)	286
11.10.2.3	round(scalar_t f)	286
11.10.2.4	sum_diff(T *begin, T *end, T *other)	286
11.10.3	Variable Documentation	286
11.10.3.1	QIWeights	286
11.10.3.2	XCorScale	286
11.10.3.3	ZLUTWeights	286
11.10.3.4	ZLUTWeights_d	286
11.11	cputrack/cpu_tracker.h File Reference	286
11.11.1	Typedef Documentation	287
11.11.1.1	pixel_t	287
11.11.2	Function Documentation	287
11.11.2.1	CreateCPUTrackerInstance(int w, int h, int xcorw)	287
11.12	cputrack/CubicBSpline.h File Reference	287
11.12.1	Macro Definition Documentation	287
11.12.1.1	CUDA_SUPPORTED_FUNC	287

11.12.2 Function Documentation	287
11.12.2.1 ComputeBSplineDerivatives(float t, T *k, T &deriv, T &deriv2)	287
11.12.2.2 ComputeBSplineWeights(float w[], float t)	288
11.12.2.3 ComputeSplineFitMaxPos(T *data, int len)	288
11.13cputrack/DebugResultCompare.h File Reference	288
11.14cputrack/dllmacros.h File Reference	288
11.14.1 Macro Definition Documentation	289
11.14.1.1 CDLL_EXPORT	289
11.14.1.2 DLL_CALLCONV	289
11.14.1.3 DLL_EXPORT	289
11.15cputrack/fastjpg.cpp File Reference	289
11.15.1 Function Documentation	289
11.15.1.1 FloatToJPEGFile(const char *name, const float *d, int w, int h)	289
11.15.1.2 ReadJPEGFile(uchar *srcbuf, int srclen, uchar **data, int *width, int *height)	290
11.15.1.3 WriteJPEGFile(uchar *data, int w, int h, const char *filename, int quality)	290
11.16cputrack/FFT2DTracker.h File Reference	292
11.17cputrack/FisherMatrix.h File Reference	292
11.18cputrack/hash_templates.h File Reference	292
11.19cputrack/kissfft.h File Reference	293
11.20cputrack/labview.h File Reference	293
11.20.1 Macro Definition Documentation	294
11.20.1.1 _STDINT_H	294
11.21cputrack/LsqQuadraticFit.h File Reference	294
11.21.1 Macro Definition Documentation	294
11.21.1.1 CUDA_SUPPORTED_FUNC	294
11.22cputrack/lv_cputrack_api.cpp File Reference	294
11.22.1 Function Documentation	295
11.22.1.1 compute_asymmetry(CPUTracker *tracker, LVArray< float > **result, int radialSteps, float *radii, float *center, uint *boundaryHit)	295
11.22.1.2 compute_com(CPUTracker *tracker, float *out)	295

11.22.1.3 compute_crp(CPUTracker *tracker, LVArray< float > **result, int radialSteps, float *radii, float *center, uint *boundaryHit, LVArray2D< float > **crpmap) . . .	296
11.22.1.4 compute_qi(CPUTracker *tracker, vector2f *position, int iterations, int radialSteps, int angularStepsPerQ, float minRadius, float maxRadius, LVArray< float > **radialweights)	296
11.22.1.5 compute_radial_profile(CPUTracker *tracker, LVArray< float > **result, int angularSteps, float *radii, float *center, uint *boundaryHit)	296
11.22.1.6 compute_xcor(CPUTracker *tracker, vector2f *position, int iterations, int profileWidth)	296
11.22.1.7 compute_z(CPUTracker *tracker, float *center, int angularSteps, int zlut_index, uint *error, LVArray< float > **profile, int *bestIndex, LVArray< float > **errorCurve)	296
11.22.1.8 create_tracker(uint w, uint h, uint xcorw)	297
11.22.1.9 destroy_tracker(CPUTracker *tracker)	297
11.22.1.10 generate_image_from_lut(LVArray2D< float > **image, LVArray2D< float > **lut, float *LUTradii, vector3f *position, float pixel_max, int useSplineInterp, int samplesPerPixel)	297
11.22.1.11 generate_test_image(LVArray2D< float > **img, float xp, float yp, float size, float photoncount)	297
11.22.1.12 get_debug_img_as_array(CPUTracker *tracker, LVArray2D< float > **pdbglImg)	298
11.22.1.13 get_ZLUT(CPUTracker *tracker, int zlutIndex, LVArray2D< float > **dst)	298
11.22.1.14 set_image_float(CPUTracker *tracker, LVArray2D< float > **pData, ErrorCluster *error)	298
11.22.1.15 set_image_from_memory(CPUTracker *tracker, LVArray2D< uchar > **pData, ErrorCluster *error)	298
11.22.1.16 set_image_u16(CPUTracker *tracker, LVArray2D< ushort > **pData, ErrorCluster *error)	298
11.22.1.17 set_image_u8(CPUTracker *tracker, LVArray2D< uchar > **pData, ErrorCluster *error)	299
11.22.1.18 set_ZLUT(CPUTracker *tracker, LVArray3D< float > **pZlut, float *radii, int angular_steps, bool useCorrelation, LVArray< float > **radialweights, bool normalize)	299
11.23 cputrack/lv_qtrk_api.h File Reference	299
11.24 cputrack/lv_queuetrk_api.cpp File Reference	300
11.24.1 Function Documentation	302
11.24.1.1 CheckImageInput(QueuedTracker *qtrk, LVArray2D< T > **data, ErrorCluster *error)	302
11.24.1.2 qtrk_build_lut_plane(QueuedTracker *qtrk, LVArray3D< float > **data, uint flags, int plane, ErrorCluster *err)	303

11.24.1.3 qtrk_compute_fisher(LVArray2D< float > **lut, QTrkSettings *cfg, vector3f *pos, LVArray2D< float > **fisherMatrix, LVArray2D< float > **inverseMatrix, vector3f *xyzVariance, int Nsamples, float maxPixelValue)	303
11.24.1.4 qtrk_compute_zlut_bias_table(QueuedTracker *qtrk, int bias_planes, LVArray2D< float > **lvresult, int smpPerPixel, int useSplineInterp, ErrorCluster *e)	303
11.24.1.5 qtrk_enable_zlut_cmpprof(QueuedTracker *qtrk, bool enable, ErrorCluster *e)	304
11.24.1.6 qtrk_finalize_lut(QueuedTracker *qtrk, ErrorCluster *e)	304
11.24.1.7 qtrk_find_beads(uint8_t *image, int pitch, int w, int h, int *smpCornerPos, int roi, float imgRelDist, float acceptance, LVArray2D< uint32_t > **output)	304
11.24.1.8 qtrk_free_all()	304
11.24.1.9 qtrk_generate_gaussian_spot(LVArray2D< float > **image, vector2f *pos, float sigma, float l0, float lbg, int applyNoise)	304
11.24.1.10 qtrk_get_computed_config(QueuedTracker *qtrk, QTrkComputedConfig *cc, ErrorCluster *err)	304
11.24.1.11 qtrk_get_debug_image(QueuedTracker *qtrk, int id, LVArray2D< float > **data, ErrorCluster *e)	305
11.24.1.12 qtrk_get_image_lut(QueuedTracker *qtrk, LVArrayND< float, 4 > **imageLUT, ErrorCluster *e)	305
11.24.1.13 qtrk_get_queue_len(QueuedTracker *qtrk, int *maxQueueLen, ErrorCluster *e)	305
11.24.1.14 qtrk_get_zlut_cmpprof(QueuedTracker *qtrk, LVArray2D< float > **output, ErrorCluster *e)	305
11.24.1.15 qtrk_set_image_lut(QueuedTracker *qtrk, LVArrayND< float, 4 > **imageLUT, LVArray3D< float > **radialZLUT, ErrorCluster *e)	305
11.24.1.16 qtrk_set_localization_mode(QueuedTracker *qtrk, uint locType, ErrorCluster *e)	306
11.24.1.17 qtrk_set_logfile_path(const char *path)	306
11.24.1.18 qtrk_set_pixel_calib(QueuedTracker *qtrk, LVArray3D< float > **offset, LV← Array3D< float > **gain, ErrorCluster *e)	306
11.24.1.19 qtrk_set_pixel_calib_factors(QueuedTracker *qtrk, float offsetFactor, float gain← Factor, ErrorCluster *e)	306
11.24.1.20 qtrk_set_zlut_bias_table(QueuedTracker *qtrk, LVArray2D< float > **biastbl, ErrorCluster *e)	307
11.24.1.21 qtrk_simulate_tracking(QueuedTracker *qtrk, int nsmp, int beadIndex, vector3f *centerPos, vector3f *range, vector3f *outBias, vector3f *outScatter, float photonsPerWell, ErrorCluster *e)	307
11.24.1.22 qtrkcuda_get_device(int device, void *info, ErrorCluster *e)	308
11.24.1.23 test_array_passing(int n, LVArray< float > **flt1D, LVArray2D< float > **flt2D, LVArray< int > **int1D, LVArray2D< int > **int2D)	308
11.24.2 Variable Documentation	308

11.24.2.1 trackerList	308
11.24.2.2 trackerListMutex	308
11.25 cputrack/lv_resultmanager_api.cpp File Reference	308
11.25.1 Function Documentation	309
11.25.1.1 rm_create(const char *file, const char *frameinfo, ResultManagerConfig *cfg, LocalizationResult *names)	309
11.25.1.2 rm_destroy(ResultManager *rm, ErrorCluster *err)	309
11.25.1.3 rm_destroy_all()	309
11.25.1.4 rm_flush(ResultManager *rm, ErrorCluster *err)	309
11.25.1.5 rm_getbeadresults(ResultManager *rm, int start, int numFrames, int bead, LocalizationResult *results, ErrorCluster *err)	310
11.25.1.6 rm_getconfig(ResultManager *rm, ResultManagerConfig *cfg, ErrorCluster *err)	310
11.25.1.7 rm_getframecounters(ResultManager *rm, int *startFrame, int *lastSaveFrame, int *endFrame, int *capturedFrames, int *localizationsDone, int *lostFrames, ErrorCluster *err)	310
11.25.1.8 rm_getresults(ResultManager *rm, int startFrame, int numFrames, LocalizationResult *results, ErrorCluster *err)	310
11.25.1.9 rm_removebead(ResultManager *rm, int bead, ErrorCluster *err)	310
11.25.1.10 rm_set_tracker(ResultManager *rm, QueuedTracker *qtrk, ErrorCluster *err)	311
11.25.1.11 rm_store_frame_info(ResultManager *rm, int frame, double timestamp, float *cols, ErrorCluster *err)	311
11.25.1.12 ValidRM(ResultManager *rm, ErrorCluster *err)	311
11.25.2 Variable Documentation	311
11.25.2.1 rm_instances	311
11.26 cputrack/memdbg.cpp File Reference	311
11.27 cputrack/memdbg.h File Reference	311
11.28 cputrack/qtrk_c_api.cpp File Reference	311
11.29 cputrack/qtrk_c_api.h File Reference	312
11.29.1 Macro Definition Documentation	314
11.29.1.1 QTrkCUDA_UseAll	314
11.29.1.2 QTrkCUDA_UseBest	314
11.29.1.3 QTrkCUDA_UseList	314
11.29.2 Typedef Documentation	314

11.29.2.1 LocMode_t	314
11.29.3 Enumeration Type Documentation	314
11.29.3.1 LocalizeModeEnum	314
11.29.3.2 QTRK_PixelDataType	315
11.30 cputrack/QueuedCPUTracker.cpp File Reference	315
11.30.1 Function Documentation	315
11.30.1.1 CreateQueuedTracker(const QTrkComputedConfig &cc)	315
11.30.1.2 SetCUDADevices(int *dev, int ndev)	315
11.31 cputrack/QueuedCPUTracker.h File Reference	315
11.32 cputrack/QueuedTracker.cpp File Reference	316
11.32.1 Macro Definition Documentation	316
11.32.1.1 WRITEVAR	316
11.32.1.2 WRITEVAR	316
11.33 cputrack/QueuedTracker.h File Reference	316
11.33.1 Macro Definition Documentation	317
11.33.1.1 BUILDLUT_BIASCORRECT	317
11.33.1.2 BUILDLUT_FOURIER	317
11.33.1.3 BUILDLUT_IMAGELUT	317
11.33.1.4 BUILDLUT_NORMALIZE	317
11.33.1.5 MIN_RADPROFILE_SMP_COUNT	317
11.33.1.6 QI_LSQFIT_NWEIGHTS	317
11.33.1.7 QI_LSQFIT_WEIGHTS	317
11.33.1.8 ZLUT_LSQFIT_NWEIGHTS	317
11.33.1.9 ZLUT_LSQFIT_WEIGHTS	317
11.33.2 Typedef Documentation	317
11.33.2.1 ImageData	317
11.33.3 Function Documentation	317
11.33.3.1 CopyImageToFloat(uchar *data, int width, int height, int pitch, QTRK_PixelData← Type pdt, float *dst)	317
11.33.3.2 CreateQueuedTracker(const QTrkComputedConfig &cc)	318
11.33.3.3 PDT_BytesPerPixel(QTRK_PixelDataType pdt)	318

11.33.3.4 SetCUDADevices(int *devices, int numdev)	318
11.34 cputrack/random_distr.h File Reference	318
11.34.1 Function Documentation	318
11.34.1.1 rand_normal()	318
11.34.1.2 rand_poisson(T lambda)	319
11.34.1.3 rand_uniform()	319
11.35 cputrack/ResultManager.cpp File Reference	319
11.35.1 Macro Definition Documentation	319
11.35.1.1 BINFILE_VERSION	319
11.36 cputrack/ResultManager.h File Reference	319
11.37 cputrack/scalar_types.h File Reference	320
11.37.1 Typedef Documentation	320
11.37.1.1 complex_t	320
11.37.1.2 scalar_t	320
11.38 cputrack/std_incl.h File Reference	320
11.38.1 Macro Definition Documentation	321
11.38.1.1 _CRT_SECURE_NO_WARNINGS	321
11.38.1.2 _CRT_SECURE_NO_WARNINGS	321
11.38.1.3 ALLOCA	321
11.38.1.4 ALLOCA_ARRAY	321
11.38.1.5 SNPRINTF	321
11.38.1.6 STRCASECMP	321
11.38.1.7 STRNCASECMP	321
11.38.1.8 VSNPRINTF	322
11.38.2 Typedef Documentation	322
11.38.2.1 uchar	322
11.38.2.2 uint	322
11.38.2.3 ulong	322
11.38.2.4 ushort	322
11.38.2.5 vector2d	322

11.38.2.6 vector2f	322
11.38.2.7 vector3d	322
11.38.2.8 vector3f	322
11.38.3 Function Documentation	322
11.38.3.1 sqrt(const vector3< T > &a)	322
11.39 cputrack/threads.h File Reference	322
11.39.1 Typedef Documentation	323
11.39.1.1 ThreadHandle	323
11.39.2 Function Documentation	323
11.39.2.1 parallel_for(int count, TF f)	323
11.40 cputrack/utils.cpp File Reference	323
11.40.1 Function Documentation	324
11.40.1.1 ApplyGaussianNoise(ImageData &img, float sigma)	324
11.40.1.2 ApplyPoissonNoise(ImageData &img, float poissonMax, float maxval)	325
11.40.1.3 ComputeBgCorrectedCOM1D(float *data, int len, float cf)	325
11.40.1.4 ComputeCRP(float *dst, int radialSteps, int angularSteps, float minradius, float maxradius, vector2f center, ImageData *img, float paddingValue, float *crpmap)	325
11.40.1.5 ComputeRadialBinWindow(int rsteps)	326
11.40.1.6 ComputeRadialProfile(float *dst, int radialSteps, int angularSteps, float minradius, float maxradius, vector2f center, ImageData *img, float mean, bool normalize)	326
11.40.1.7 CopyImageToFloat(uchar *data, int width, int height, int pitch, QTRK_PixelData< Type pdt, float *dst)	327
11.40.1.8 dbgout(const std::string &s)	327
11.40.1.9 dbgprintf(const char *fmt,...)	327
11.40.1.10 dbgsetlogfile(const char *path)	328
11.40.1.11 file_ext(const char *f)	328
11.40.1.12 GenerateGaussianSpotImage(ImageData *img, vector2f pos, float sigma, float l0, float lbg)	328
11.40.1.13 GenerateImageFromLUT(ImageData *image, ImageData *zlut, float minradius, float maxradius, vector3f pos, bool splineInterp, int oversampleSubdiv)	328
11.40.1.14 GenerateTestImage(ImageData &img, float xp, float yp, float size, float SNratio)	329
11.40.1.15 GetCurrentOutputPath(bool ext)	329
11.40.1.16 GetDirectoryFromPath(std::string fullname)	330

11.40.1.17GetFormattedTimeString(char *output)	330
11.40.1.18GetLocalModuleFilename()	330
11.40.1.19GetLocalModulePath()	330
11.40.1.20GetPreciseTime()	330
11.40.1.21NearestPowerOf2(int v)	331
11.40.1.22NearestPowerOf3(int v)	331
11.40.1.23NormalizeRadialProfile(scalar_t *prof, int rsteps)	331
11.40.1.24NormalizeZLUT(float *zlut, int numBeads, int planes, int radialsteps)	331
11.40.1.25ReadCSV(const char *filename, char sep)	332
11.40.1.26ReadJPEGFile(const char *fn)	332
11.40.1.27ReadLUTFile(const char *lutfile)	332
11.40.1.28ReadToByteBuffer(const char *filename)	333
11.40.1.29ReadVector3CSV(const char *file, char sep)	333
11.40.1.30SPrintf(const char *fmt,...)	333
11.40.1.31sq(float x)	333
11.40.1.32WriteArrayAsCSVRow(const char *file, float *d, int len, bool append)	333
11.40.1.33WriteComplexImageAsCSV(const char *file, std::complex< float > *d, int w, int h, const char *labels[])	334
11.40.1.34WriteImageAsCSV(const char *file, float *d, int w, int h, const char *labels[])	334
11.40.1.35WriteToLog(const char *str)	334
11.40.1.36WriteTrace(std::string filename, vector3f *results, int nResults)	335
11.40.1.37WriteVectorAsCSVRow(const char *file, std::vector< float > d, bool append)	335
11.40.2 Variable Documentation	335
11.40.2.1 logFilename	335
11.41 cputrack/utils.h File Reference	335
11.41.1 Typedef Documentation	337
11.41.1.1 ImageData	337
11.41.1.2 ImageDatad	337
11.41.2 Function Documentation	337
11.41.2.1 ApplyGaussianNoise(ImageData &img, float sigma)	337
11.41.2.2 ApplyPoissonNoise(ImageData &img, float poissonMax, float maxValue=255)	337

11.41.2.3 ComputeBgCorrectedCOM1D(float *data, int len, float cf=2.0f)	338
11.41.2.4 ComputeCRP(float *dst, int radialSteps, int angularSteps, float minradius, float maxradius, vector2f center, ImageData *src, float mean, float *crpmap=0)	338
11.41.2.5 ComputeRadialBinWindow(int rsteps)	339
11.41.2.6 ComputeRadialProfile(float *dst, int radialSteps, int angularSteps, float minradius, float maxradius, vector2f center, ImageData *src, float mean, bool normalize)	339
11.41.2.7 ComputeStdDev(T *data, int len)	339
11.41.2.8 dbgout(const std::string &s)	340
11.41.2.9 dbgprintf(const char *fmt,...)	340
11.41.2.10 dbgsetlogfile(const char *path)	340
11.41.2.11 DeleteAllElems(T &c)	340
11.41.2.12 erf(T x)	340
11.41.2.13 file_ext(const char *)	341
11.41.2.14 FloatToJPEGFile(const char *name, const float *d, int w, int h)	341
11.41.2.15 floatToNormalizedInt(T *dst, const float *src, uint w, uint h, T maxValue)	341
11.41.2.16 floatToNormalizedInt(const float *src, uint w, uint h, T maxValue)	341
11.41.2.17 GenerateGaussianSpotImage(ImageData *img, vector2f pos, float sigma, float l0, float lbg)	341
11.41.2.18 GenerateImageFromLUT(ImageData *image, ImageData *zlut, float minradius, float maxradius, vector3f pos, bool useSplineInterp=true, int ovs=4)	342
11.41.2.19 GenerateTestImage(ImageData &img, float xp, float yp, float size, float MaxPhotons)	342
11.41.2.20 GetCurrentOutputPath(bool ext=true)	343
11.41.2.21 GetDirectoryFromPath(std::string fullpath)	343
11.41.2.22 GetFormattedTimeString(char *output)	343
11.41.2.23 GetLocalModuleFilename()	343
11.41.2.24 GetLocalModulePath()	344
11.41.2.25 GetPreciseTime()	344
11.41.2.26 Interpolate(T *image, int width, int height, float x, float y, bool *outside=0)	344
11.41.2.27 Interpolate1D(T *d, int len, float x)	344
11.41.2.28 Interpolate1D(const std::vector<T> &d, float x)	344
11.41.2.29 sNAN(const T &v)	344
11.41.2.30 Lerp(T a, T b, float x)	345

11.41.2.31	NearestPowerOf2(int v)	345
11.41.2.32	NearestPowerOf3(int v)	345
11.41.2.33	normalize(TPixel *d, uint w, uint h)	345
11.41.2.34	NormalizeRadialProfile(float *prof, int rsteps)	345
11.41.2.35	NormalizeZLUT(float *zlut, int numLUTs, int planes, int radialsteps)	346
11.41.2.36	qselect(T *data, int start, int end, int k)	346
11.41.2.37	ReadCSV(const char *filename, char sep='t')	346
11.41.2.38	ReadJPEGFile(const char *fn)	347
11.41.2.39	ReadJPEGFile(uchar *srcbuf, int srclen, uchar **data, int *width, int *height)	347
11.41.2.40	ReadLUTFile(const char *lutfile)	348
11.41.2.41	ReadToByteBuffer(const char *filename)	348
11.41.2.42	ReadVector3CSV(const char *file, char sep='t')	348
11.41.2.43	Sprintf(const char *fmt,...)	349
11.41.2.44	StdDeviation(T *start, T *end)	349
11.41.2.45	WriteArrayAsCSVRow(const char *file, float *d, int len, bool append)	349
11.41.2.46	WriteComplexImageAsCSV(const char *file, std::complex< float > *d, int w, int h, const char *labels[] = 0)	349
11.41.2.47	WriteImageAsCSV(const char *file, float *d, int w, int h, const char *labels[] = 0)	350
11.41.2.48	WriteJPEGFile(uchar *data, int w, int h, const char *filename, int quality)	350
11.41.2.49	WriteJPEGFile(const char *name, const ImageData &img)	351
11.41.2.50	WriteTrace(std::string file, vector3f *results, int nResults)	351
11.41.2.51	WriteVectorAsCSVRow(const char *file, std::vector< float > d, bool append)	352
11.42	cudatrack-test/test.cu File Reference	352
11.42.1	Function Documentation	353
11.42.1.1	BasicQTrkTest()	353
11.42.1.2	BasicQTrkTest_RM()	354
11.42.1.3	BenchmarkParams()	355
11.42.1.4	BuildZLUT(std::string folder, outputter *output)	356
11.42.1.5	check_arg(const std::vector< std::string > &args, const char *name, T *param)	357
11.42.1.6	check_strarg(const std::vector< std::string > &args, const char *name, std::string *param)	357

11.42.1.7 CmdLineRun(int argc, char *argv[])	357
11.42.1.8 CompareAccuracy(const char *lutfile)	359
11.42.1.9 compute(int idx, float *buf, int s)	360
11.42.1.10emptyKernel()	360
11.42.1.11getPath(const char *file)	360
11.42.1.12listDevices()	360
11.42.1.13main(int argc, char *argv[])	361
11.42.1.14mul_conjugate(float2 a, float2 b)	361
11.42.1.15NearestPowerOfTwo(int v)	361
11.42.1.16ProfileSpeedVsROI(LocalizeModeEnum locMode, const char *outputcsv, bool haveZLUT, int qi_iterations)	362
11.42.1.17QICompare(const char *lutfile)	362
11.42.1.18QTrkCompareTest()	363
11.42.1.19ShowCUDAError()	364
11.42.1.20SimpleKernel(int N, float *a)	364
11.42.1.21SmallestPowerOfTwo(int minval)	365
11.42.1.22SpeedCompareTest(int w, LocalizeModeEnum locMode, bool haveZLUT, int qi_iterations=5)	365
11.42.1.23SpeedTest(const QTrkSettings &cfg, QueuedTracker *qtrk, int count, bool haveZLUT, LocMode_t locType, float *scheduleTime, bool gainCorrection=false)	365
11.42.1.24TestAsync()	366
11.42.1.25TestBenchmarkLUT()	367
11.42.1.26TestGauss2D(bool calib)	367
11.42.1.27TestRadialLUTGradientMethod()	367
11.42.1.28TestSharedMem()	367
11.42.1.29testWithGlobal(int n, int s, float *result, float *buf)	368
11.42.1.30testWithShared(int n, int s, float *result)	368
11.42.2 Variable Documentation	368
11.42.2.1 cmp_cpu_qi_fft_out	368
11.42.2.2 cmp_cpu_qi_prof	368
11.42.2.3 cmp_gpu_qi_fft_out	368
11.42.2.4 cmp_gpu_qi_prof	368

11.42.2.5 cudaSharedMem	368
11.43cudatrack/cudalmageList.h File Reference	368
11.44cudatrack/gpu_utils.h File Reference	368
11.44.1 Macro Definition Documentation	369
11.44.1.1 CUBOTH	369
11.44.1.2 CUDA_SUPPORTED_FUNC	369
11.44.2 Function Documentation	369
11.44.2.1 CheckCUDAError(cufftResult_t err)	369
11.44.2.2 CheckCUDAError(cudaError_t err)	369
11.44.2.3 CheckCUDAError()	370
11.44.2.4 DbgCopyResult(device_vec< float2 > src, std::vector< std::complex< float > > &dst)	370
11.44.2.5 DbgCopyResult(device_vec< float > src, std::vector< float > &dst)	370
11.44.2.6 dbgCUDAErrorCheck(cudaError_t e)	370
11.44.2.7 DbgOutputVectorToFile(std::string loc, device_vec< float > &src, bool append)	370
11.44.2.8 outputTotalGPUMemUse(std::string info="")	370
11.45cudatrack/ImageSampler.h File Reference	370
11.45.1 Typedef Documentation	371
11.45.1.1 ImageSampler_Tex	371
11.45.2 Function Documentation	371
11.45.2.1 image_sampler_texture_linear(0, cudaFilterModeLinear)	371
11.45.2.2 image_sampler_texture_nearest(0, cudaFilterModePoint)	371
11.46cudatrack/Kernels.h File Reference	371
11.46.1 Function Documentation	372
11.46.1.1 ApplyOffsetGain(BaseKernelParams kp, cudalmageListf calib_gain, cudaImageListf calib_offset, float gainFactor, float offsetFactor)	372
11.46.1.2 BgCorrectedCOM(int idx, cudalmageListf images, float correctionFactor, float *pMean)	372
11.46.1.3 BgCorrectedCOM(int count, cudalmageListf images, float3 *d_com, float bgCorrectionFactor, float *d_imgmeans)	373
11.46.1.4 ForceCUDAKernelstoLoad()	373
11.46.1.5 G2MLE_Compute(BaseKernelParams kp, float sigma, int iterations, float3 *initial, float3 *positions, float *l_bg, float *l_0)	373

11.46.1.6 ImageLUT_Sample(BaseKernelParams kp, float2 ilut_scale, float3 *positions, typename TImageLUT::KernelParams lut)	374
11.46.1.7 interpolate(T a, T b, float x)	374
11.46.1.8 ZLUT_ComputeProfileMatchScores(int njobs, ZLUTParams params, float *profiles, float *compareScoreBuf, LocalizationParams *locParams)	375
11.46.1.9 ZLUT_ComputeZ(int njobs, ZLUTParams params, float3 *positions, float *compareScoreBuf)	375
11.46.1.10 ZLUT_NormalizeProfiles(int njobs, ZLUTParams params, float *profiles)	375
11.46.1.11 ZLUT_ProfilesToZLUT(int njobs, cudalImageListf images, ZLUTParams params, float3 *positions, LocalizationParams *locParams, float *profiles)	376
11.46.1.12 ZLUT_RadialProfileKernel(int njobs, cudalImageListf images, ZLUTParams params, float3 *positions, float *profiles, float *means)	376
11.46.2 Variable Documentation	376
11.46.2.1 image_lut_surface	376
11.47 cudatrack/QI.h File Reference	376
11.47.1 Typedef Documentation	377
11.47.1.1 qicomplex_t	377
11.47.1.2 qivalue_t	377
11.48 cudatrack/QI_impl.h File Reference	377
11.48.1 Function Documentation	377
11.48.1.1 ComputeQuadrantProfile(cudalImageListf &images, int idx, float *dst, const QI Params ¶ms, int quadrant, float2 center, float mean, int angularSteps)	377
11.48.1.2 QI_ComputeAxisOffset(cufftComplex *autoconv, int fftlen, float *shiftbuf)	378
11.48.1.3 QI_ComputeProfile(BaseKernelParams kp, float3 *positions, float *quadrants, float2 *profiles, float2 *reverseProfiles, const QIParams qiParams, float *d_radialweights, int angularSteps)	378
11.48.1.4 QI_ComputeQuadrants(BaseKernelParams kp, float3 *positions, float *dst_quadrants, const QIParams *params, int angularSteps)	379
11.48.1.5 QI_MultiplyWithConjugate(int n, cufftComplex *a, cufftComplex *b)	379
11.48.1.6 QI_OffsetPositions(int njobs, float3 *current, float3 *dst, cufftComplex *autoconv, int fftLength, float2 *offsets, float pixelsPerProfLen, float *shiftbuf)	380
11.48.1.7 QI_QuadrantsToProfiles(BaseKernelParams kp, float *quadrants, float2 *profiles, float2 *reverseProfiles, float *d_radialweights, const QIParams *params)	380
11.49 cudatrack/QueuedCUDATracker.cu File Reference	381
11.49.1 Macro Definition Documentation	382

11.49.1.1 TRK_PROFILE	382
11.49.2 Function Documentation	382
11.49.2.1 AddProfilesToZLUT(float *d_src, int nbeads, int radialsteps, int plane, cudaImageListf zlut)	382
11.49.2.2 checksum(T *data, int elemsize, int numelem, const char *name)	382
11.49.2.3 CreateQueuedTracker(const QTrkComputedConfig &cc)	382
11.49.2.4 GetBestCUDADevice()	382
11.49.2.5 SetCUDADevices(int *dev, int numdev)	383
11.49.3 Variable Documentation	383
11.49.3.1 cudaDeviceList	383
11.50 cudatrack/QueuedCUDATracker.h File Reference	383
11.50.1 Typedef Documentation	383
11.50.1.1 cudalmageListf	383
11.51 cudatrack/simplefft.h File Reference	383
11.51.1 Macro Definition Documentation	384
11.51.1.1 SFFT_BOTH	384
Index	385

Chapter 1

Queued Tracker software

1.1 Introduction

[QueuedTracker](#), or QTrk in short, is an API that facilitates the 3 dimensional subpixel tracking of a magnetic bead in a Magnetic Tweezers (MT) setup. The code found here generates a general purpose library in the form of a DLL that can be used from either .NET applications or LabVIEW. The [LabVIEW GUI and hardware control](#), which is not included in this documentation, has been created very specifically for the setups as designed and used in the [Nynke Dekker lab](#).

1.2 History

The MT setups are homebuilt devices for biological single molecule measurements. They have evolved over the years and so has the need for the related software. Considering the framerate and number of pixels of state of the art cameras involved in the setups, the requirements with regards to data handling are now very steep. A measurement running for a few hours generates hundreds of gigabytes worth of images. As such, the need arose to do the image analysis in real-time and fast. This software was created to provide precisely that. To ensure high speed data analysis, multiple algorithms have been implemented in multithreaded CPU and GPU (through CU↔DA) implementations, with a scheduling shell ([QueuedCPUTracker](#) and [QueuedCUDATracker](#)) and separate data gathering and saving thread ([ResultManager](#)).

1.3 Implementation

The goal is to find a 3 dimensional position of a bead from a microscopic image. A typical image of a single bead is displayed below: To perform the tracking, specific algorithms exist and have been implemented. Currently the available options are:

Algorithm	Dimensions	CPU	CUDA	Reference	Notes
Starting point		QueuedCPUTracker::ProcessJob()	QueuedCUDATracker::ExecuteBatch()		Functions from which the algorithms are called dependent on settings.

Algorithm	Dimensions	CPU	CUDA	Reference	Notes
Center of Mass (COM)	x,y	CPUTracker:: ComputeMean AndCOM()	BgCorrectedC OM()		Always executed for first guess.
1D Cross Correlation (XCor1D)	x,y	CPUTracker:: ComputeXCor Interpolated()	Not implemented		
Quadrant Interpolation (QI)	x,y	CPUTracker:: ComputeQI()	QI, QI::Execute()		Recommended algorithm. Optimized for speed and accuracy.
2D Gaussian fit	x,y	CPUTracker:: Compute2D GaussianMLE()	G2MLE_ Compute()		
Lookup table (LUT)	z	CPUTracker:: ComputeRadial Profile() CPUTracker:: LUTProfile Compare()	ZLUT_Radial ProfileKernel() ZLUT_ ComputeZ()		Only available method for z localization.

1.4 Required software

To be able to compile the DLLs, you need:

- Visual Studio (2010)
- CUDA (6.5)

To be able to use the CUDA DLLs, the cudart32_65.dll and cufft32_65.dll (or 64 bit versions if compiled with that) need to be known and accessible by the system, i.e. they need to be in the same folder. We are currently limited to CUDA 6.5 at the highest due to the fact that we have a 32 bit LabVIEW version on setups and 32 bit cuFFT has been deprecated in newer CUDA versions.

1.5 Credits

Initial work by Jelmer Cnossen. Maintenance, testing, documentation and improvements by Jordi Wassenburg.

Chapter 2

Todo List

Member [QTrkSettings::testRun](#)

Add to LabVIEW cluster.

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

LabVIEW datatypes and helper functions	19
API - LabVIEW	23
API - C	29

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

BeadFinder	33
kissfft_utils	34
qtrk	34
sfft	35

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Atomic< T >	37
Atomic< bool >	37
BaseKernelParams	38
BenchmarkLUT	39
LsqSqQuadFit< T >::Coeff	44
sfft::complex< T >	45
ComputeMaxInterp< T, numPts >	48
BeadFinder::Config	49
CPUTracker	50
CUDADeviceInfo	68
cudaImageList< T >	68
cudaImageList< float >	68
QueuedCUDATracker::Device	75
device_vec< T >	77
device_vec< float >	77
device_vec< float2 >	77
device_vec< float3 >	77
device_vec< LocalizationParams >	77
Ql::DeviceInstance	81
ErrorCluster	81
CPUTracker::FFT2D	82
FFT2DTracker	83
ResultManager::FrameCounters	85
ResultManager::FrameResult	86
CPUTracker::Gauss2DResult	86
Threads::Handle	87
hash_map	
qtrk::hash_map< TKey, T >	88
hash_set	
qtrk::hash_set< T >	88
Image4DCudaArray< T >	88
Image4DMemory< T >	93
ImageLUTConfig	96
ImageSampler_InterpolatedTexture	97
ImageSampler_MemCopy	98

ImageSampler_SimpleTextureRead	99
QueuedCPUTracker::Job	100
Image4DCudaArray< T >::KernelInst	101
Image4DMemory< T >::KernelParams	102
QueuedCUDATracker::KernelProfileTime	103
kissfft< T_Scalar, T_traits >	104
kissfft< float >	104
kissfft< scalar_t >	104
LocalizationJob	110
LocalizationParams	112
LocalizationResult	112
LsqSqQuadFit< T >	114
LVArray< T >	117
LVArray2D< T >	117
LVArray3D< T >	118
LVArrayND< T, N >	119
LVDataType< T >	120
LVDataType< double >	120
LVDataType< float >	120
LVDataType< int16_t >	121
LVDataType< int32_t >	121
LVDataType< int64_t >	122
LVDataType< int8_t >	122
LVDataType< std::complex< double > >	123
LVDataType< std::complex< float > >	123
LVDataType< uint16_t >	124
LVDataType< uint32_t >	124
LVDataType< uint64_t >	125
LVDataType< uint8_t >	125
Matrix3X3	126
MeasureTime	129
Threads::Mutex	130
my_error_mgr	132
outputter::outputModes	132
outputter	133
PathSeparator	135
pinned_array< T, flags >	136
pinned_array< float >	136
pinned_array< float3 >	136
pinned_array< LocalizationParams >	136
BeadFinder::Position	140
QI	140
QIPParams	144
QTrkSettings	148
QTrkComputedConfig	145
QueuedTracker	188
QueuedCPUTracker	153
QueuedCUDATracker	171
ResultFile	196
BinaryResultFile	41
TextResultFile	217
ResultManager	197
ResultManagerConfig	205
ROIPosition	205
RunTrackerResults	206
SampleFisherMatrix	207
ScopedCPUProfiler	210

SpeedAccResult	210
SpeedInfo	211
QueuedCUDATracker::Stream	212
Ql::StreamInstance	215
QueuedCPUTracker::Thread	218
ThreadPool< TWorkItem, TFunctor >	219
Threads	222
TImageData< T >	224
TImageData< float >	224
CImageData	43
kissfft_utils::traits< T_scalar >	227
vector2< T >	228
vector2< float >	228
vector3< T >	229
vector3< float >	229
XCor1DBuffer	233
ZLUTParams	234

Chapter 6

Class Index

6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Atomic< T >	37
BaseKernelParams	38
BenchmarkLUT	39
BinaryResultFile	41
CImageData	43
LsqSqQuadFit< T >::Coeff	44
sfft::complex< T >	45
ComputeMaxInterp< T, numPts >	48
BeadFinder::Config	49
CPUTracker	50
CUDADeviceInfo	68
cudaImageList< T >	68
QueuedCUDATracker::Device	75
device_vec< T >	77
QI::DeviceInstance	81
ErrorCluster	81
CPUTracker::FFT2D	82
FFT2DTracker	83
ResultManager::FrameCounters	85
ResultManager::FrameResult	86
CPUTracker::Gauss2DResult	86
Threads::Handle	87
qtrk::hash_map< TKey, T >	88
qtrk::hash_set< T >	88
Image4DCudaArray< T >	88
Image4DMemory< T >	93
ImageLUTConfig	96
ImageSampler_InterpolatedTexture	97
ImageSampler_MemCopy	98
ImageSampler_SimpleTextureRead	99
QueuedCPUTracker::Job	100
Image4DCudaArray< T >::KernelInst	101
Image4DMemory< T >::KernelParams	102
QueuedCUDATracker::KernelProfileTime	103
kissfft< T_Scalar, T_traits >	104

LocalizationJob	
Structure for region of interest metadata	110
LocalizationParams	112
LocalizationResult	
Structure for job results	112
LsqSqQuadFit< T >	114
LVArray< T >	117
LVArray2D< T >	117
LVArray3D< T >	118
LVArrayND< T, N >	119
LVDataType< T >	120
LVDataType< double >	120
LVDataType< float >	120
LVDataType< int16_t >	121
LVDataType< int32_t >	121
LVDataType< int64_t >	122
LVDataType< int8_t >	122
LVDataType< std::complex< double > >	123
LVDataType< std::complex< float > >	123
LVDataType< uint16_t >	124
LVDataType< uint32_t >	124
LVDataType< uint64_t >	125
LVDataType< uint8_t >	125
Matrix3X3	126
MeasureTime	129
Threads::Mutex	130
my_error_mngr	132
outputter::outputModes	132
outputter	133
PathSeparator	135
pinned_array< T, flags >	136
BeadFinder::Position	140
QI	140
QIParams	144
QTrkComputedConfig	
Structure for derived settings computed from base settings in QTrkSettings	145
QTrkSettings	
Structure for the settings used by the algorithms implemented in QueuedTracker	148
QueuedCPUTracker	
CPU implementation of the QueuedTracker interface	153
QueuedCUDATracker	
CUDA implementation of the QueuedTracker interface	171
QueuedTracker	
Abstract tracker interface, implemented by QueuedCUDATracker and QueuedCPUTracker	188
ResultFile	196
ResultManager	197
ResultManagerConfig	205
ROIPosition	
Struct used to define the top-left corner position of an ROI within a frame. ROI is [x .. x+w ; y .. y+h]	205
RunTrackerResults	206
SampleFisherMatrix	207
ScopedCPUProfiler	210
SpeedAccResult	210
SpeedInfo	211
QueuedCUDATracker::Stream	212
QI::StreamInstance	215
TextResultFile	217

QueuedCPUTracker::Thread	218
ThreadPool< TWorkItem, TFunctor >	219
Threads	222
TImageData< T >	224
kissfft_utils::traits< T_scalar >	227
vector2< T >	228
vector3< T >	229
XCor1DBuffer	233
ZLUTParams	234

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

cputrack-test/Benchmark.cpp	237
cputrack-test/main.cpp	240
cputrack-test/SharedTests.h	266
cputrack-test/testutils.cpp	273
cputrack-test/testutils.h	277
cputrack/BeadFinder.cpp	281
cputrack/BeadFinder.h	284
cputrack/BenchmarkLUT.cpp	284
cputrack/BenchmarkLUT.h	284
cputrack/cpu_tracker.cpp	285
cputrack/cpu_tracker.h	286
cputrack/CubicBSpline.h	287
cputrack/DebugResultCompare.h	288
cputrack/dllmacros.h	288
cputrack/fastjpg.cpp	289
cputrack/FFT2DTracker.h	292
cputrack/FisherMatrix.h	292
cputrack/hash_templates.h	292
cputrack/kissfft.h	293
cputrack/labview.h	293
cputrack/LsqQuadraticFit.h	294
cputrack/lv_cputrack_api.cpp	294
cputrack/lv_qtrk_api.h	299
cputrack/lv_queuetrk_api.cpp	300
cputrack/lv_resultmanager_api.cpp	308
cputrack/memdbg.cpp	311
cputrack/memdbg.h	311
cputrack/qtrk_c_api.cpp	311
cputrack/qtrk_c_api.h	312
cputrack/QueuedCPUTracker.cpp	315
cputrack/QueuedCPUTracker.h	315
cputrack/QueuedTracker.cpp	316
cputrack/QueuedTracker.h	316
cputrack/random_distr.h	318
cputrack/ResultManager.cpp	319

cputrack/ ResultManager.h	319
cputrack/ scalar_types.h	320
cputrack/ std_incl.h	320
cputrack/ threads.h	322
cputrack/ utils.cpp	323
cputrack/ utils.h	335
cudatrack-test/ test.cu	352
cudatrack/ cudalImageList.h	368
cudatrack/ gpu_utils.h	368
cudatrack/ ImageSampler.h	370
cudatrack/ Kernels.h	371
cudatrack/ QI.h	376
cudatrack/ QI_impl.h	377
cudatrack/ QueuedCUDATracker.cu	381
cudatrack/ QueuedCUDATracker.h	383
cudatrack/ simplefft.h	383

Chapter 8

Module Documentation

8.1 LabVIEW datatypes and helper functions

Definitions of datatypes and helper functions required for communication with LabVIEW.

Classes

- struct `ErrorCluster`
- struct `LVArray< T >`
- struct `LVArray2D< T >`
- struct `LVArray3D< T >`
- struct `LVArrayND< T, N >`
- struct `LVDataType< T >`
- struct `LVDataType< float >`
- struct `LVDataType< double >`
- struct `LVDataType< int8_t >`
- struct `LVDataType< int16_t >`
- struct `LVDataType< int32_t >`
- struct `LVDataType< int64_t >`
- struct `LVDataType< uint8_t >`
- struct `LVDataType< uint16_t >`
- struct `LVDataType< uint32_t >`
- struct `LVDataType< uint64_t >`
- struct `LVDataType< std::complex< float > >`
- struct `LVDataType< std::complex< double > >`

Typedefs

- typedef `LVArray< float > ** ppFloatArray`
- typedef `LVArray2D< float > ** ppFloatArray2`

Functions

- template<typename T >
void **ResizeLVArray2D** (**LVArray2D**< T > **&d, int rows, int cols)
- template<typename T >
void **ResizeLVArray3D** (**LVArray3D**< T > **&d, int depth, int rows, int cols)
- template<typename T , int N>
void **ResizeLVArray** (**LVArrayND**< T, N > **&d, int *dims)
- template<typename T >
void **ResizeLVArray** (**LVArray**< T > **&d, int elems)
- void **ArgumentErrorMsg** (**ErrorCluster** *e, const std::string &msg)
- void **SetLVString** (LStrHandle str, const char *text)
- MgErr **FillErrorCluster** (MgErr err, const char *message, **ErrorCluster** *error)
- std::vector< std::string > **LVGetStringArray** (int count, LStrHandle *str)
- bool **ValidateTracker** (**QueuedTracker** *tracker, **ErrorCluster** *e, const char *funcname)

8.1.1 Detailed Description

Definitions of datatypes and helper functions required for communication with LabVIEW.

8.1.2 Typedef Documentation

8.1.2.1 **typedef LVArray<float>** ppFloatArray**

8.1.2.2 **typedef LVArray2D<float>** ppFloatArray2**

8.1.3 Function Documentation

8.1.3.1 **void ArgumentErrorMsg (ErrorCluster * e, const std::string & msg)**

```
61 {
62     FillErrorCluster(mgArgErr, msg.c_str(), e);
63 }
```

8.1.3.2 **MgErr FillErrorCluster (MgErr err, const char * message, ErrorCluster * error)**

```
50 {
51     if (err)
52     {
53         error->status = LVBooleanTrue;
54         error->code = err;
55         SetLVString ( error->message, message );
56     }
57     return err;
58 }
```

8.1.3.3 `std::vector<std::string> LVGetStringArray (int count, LStrHandle * str)`

```

38 {
39     std::vector<std::string> result(count);
40
41     for (int x=0;x<count;x++) {
42         uChar *val = LHStrBuf(str[x]);
43         int len = LHStrLen (str[x]);
44         result[x]=std::string((const char*)val, (size_t)len );
45     }
46     return result;
47 }
```

8.1.3.4 `template<typename T , int N> void ResizeLVArray (LVArrayND<T,N> **& d, int * dims)`

```

107 {
108     for (int i=0;i<N;i++)
109         (*d)->dimSizes[i]=dims[i];
110     NumericArrayResize(LVDataType<T>::code, N, (UHandle*)&d, sizeof(T)*(*d)->
111     numElem());
111 }
```

8.1.3.5 `template<typename T > void ResizeLVArray (LVArray<T> **& d, int elems)`

```

114 {
115     if (NumericArrayResize(LVDataType<T>::code, 1, (UHandle*)&d, sizeof(T)*elems) !=
116         mgNoErr)
117         throw std::runtime_error( SPrintf("NumericArrayResize(1D array, %d) returned error.", elems)
118     );
118 }
```

8.1.3.6 `template<typename T > void ResizeLVArray2D (LVArray2D<T> **& d, int rows, int cols)`

```

89 {
90     if (NumericArrayResize(LVDataType<T>::code, 2, (UHandle*)&d, sizeof(T)*rows*cols) !=
91         mgNoErr)
92         throw std::runtime_error( SPrintf("NumericArrayResize(2D array, %d, %d) returned error.", rows,cols));
93     (*d)->dimSizes[0] = rows;
94     (*d)->dimSizes[1] = cols;
```

8.1.3.7 `template<typename T > void ResizeLVArray3D (LVArray3D<T> **& d, int depth, int rows, int cols)`

```

97 {
98     if (NumericArrayResize(LVDataType<T>::code, 3, (UHandle*)&d, sizeof(T)*rows*cols*
99         depth) != mgNoErr)
100         throw std::runtime_error( SPrintf("NumericArrayResize(3D array, %d, %d, %d) returned error."
101         , depth,rows,cols));
102     (*d)->dimSizes[0] = depth;
103     (*d)->dimSizes[1] = rows;
104     (*d)->dimSizes[2] = cols;
```

8.1.3.8 void SetLVString (LStrHandle str, const char * text)

```
27 {
28     int msglen = strlen(text);
29     MgErr err = NumericArrayResize(uB, 1, (UHandle*)&str, msglen);
30     if (!err)
31     {
32         MoveBlock(text, LStrBuf(*str), msglen);
33         LStrLen(*str) = msglen;
34     }
35 }
```

8.1.3.9 bool ValidateTracker (QueuedTracker * tracker, ErrorCluster * e, const char * funcname)

```
66 {
67     if (std::find(trackerList.begin(), trackerList.end(), tracker)==
68         trackerList.end()) {
69         ArgumentErrorMsg(e, SPrintf("QTrk C++ function %s called with invalid
70         tracker pointer: %p", funcname, tracker));
71         return false;
72     }
73     return true;
74 }
```

8.2 API - LabVIEW

API functions available to LabVIEW.

Classes

- struct [CUDADeviceInfo](#)

Enumerations

- enum [QueueFrameFlags](#) { QFF_Force32Bit = 0x7fffffff }

Functions

- [CDLL_EXPORT void DLL_CALLCONV qtrk_set_ZLUT](#) (QueuedTracker *tracker, LVArray3D< float > **pZlut, LVArray< float > **zcmpWindow, int normalize, ErrorCluster *e)
- [CDLL_EXPORT void DLL_CALLCONV qtrk_get_ZLUT](#) (QueuedTracker *tracker, LVArray3D< float > **pzlut, ErrorCluster *e)
- [CDLL_EXPORT QueuedTracker *DLL_CALLCONV qtrk_create](#) (QTrkSettings *settings, LStrHandle warnings, ErrorCluster *e)
- [CDLL_EXPORT void DLL_CALLCONV qtrk_destroy](#) (QueuedTracker *qtrk, ErrorCluster *error)
- [CDLL_EXPORT void DLL_CALLCONV qtrk_queue_u16](#) (QueuedTracker *qtrk, ErrorCluster *error, LVArray2D< ushort > **data, const LocalizationJob *jobInfo)
- [CDLL_EXPORT void DLL_CALLCONV qtrk_queue_u8](#) (QueuedTracker *qtrk, ErrorCluster *error, LVArray2D< uchar > **data, const LocalizationJob *jobInfo)
- [CDLL_EXPORT void DLL_CALLCONV qtrk_queue_float](#) (QueuedTracker *qtrk, ErrorCluster *error, LVArray2D< float > **data, const LocalizationJob *jobInfo)
- [CDLL_EXPORT void DLL_CALLCONV qtrk_queue_pitchedmem](#) (QueuedTracker *qtrk, uchar *data, int pitch, uint pdt, const LocalizationJob *jobInfo)
- [CDLL_EXPORT void DLL_CALLCONV qtrk_queue_array](#) (QueuedTracker *qtrk, ErrorCluster *error, LVArray2D< uchar > **data, uint pdt, const LocalizationJob *jobInfo)
- [CDLL_EXPORT uint qtrk_read_timestamp](#) (uchar *image, int w, int h)
- [CDLL_EXPORT uint DLL_CALLCONV qtrk_queue_frame](#) (QueuedTracker *qtrk, uchar *image, int pitch, int w, int h, uint pdt, ROIPosition *pos, int numROI, const LocalizationJob *pJobInfo, QueueFrameFlags flags, ErrorCluster *e)
- [CDLL_EXPORT void DLL_CALLCONV qtrk_clear_results](#) (QueuedTracker *qtrk, ErrorCluster *e)
- [CDLL_EXPORT int DLL_CALLCONV qtrk_resultcount](#) (QueuedTracker *qtrk, ErrorCluster *e)
- [CDLL_EXPORT void DLL_CALLCONV qtrk_flush](#) (QueuedTracker *qtrk, ErrorCluster *e)
- [CDLL_EXPORT int DLL_CALLCONV qtrk_get_results](#) (QueuedTracker *qtrk, LocalizationResult *results, int maxResults, int sortByID, ErrorCluster *e)
- [CDLL_EXPORT int DLL_CALLCONV qtrk_idle](#) (QueuedTracker *qtrk, ErrorCluster *e)
- [CDLL_EXPORT void DLL_CALLCONV qtrk_generate_image_from_lut](#) (LVArray2D< float > **image, LVArray2D< float > **lut, float *LUTradii, vector2f *position, float z, float M, float sigma_noise)
- [CDLL_EXPORT void DLL_CALLCONV qtrk_dump_memleaks](#) ()
- [CDLL_EXPORT void qtrk_get_profile_report](#) (QueuedTracker *qtrk, LStrHandle str)
- [CDLL_EXPORT int DLL_CALLCONV qtrkcuda_device_count](#) (ErrorCluster *e)
- [CDLL_EXPORT void DLL_CALLCONV qtrkcuda_set_device_list](#) (LVArray< int > **devices)

8.2.1 Detailed Description

API functions available to LabVIEW.

These DLLs are compiled by the */vcputrack* and */vcudatrack* projects.

8.2.2 Enumeration Type Documentation

8.2.2.1 enum QueueFrameFlags

Enumerator

QFF_Force32Bit

```
28
29     QFF_Force32Bit = 0x7fffffff
30 
```

8.2.3 Function Documentation

8.2.3.1 CDLL_EXPORT void DLL_CALLCONV qtrk_clear_results (QueuedTracker * qtrk, ErrorCluster * e)

```
347 {
348     if (ValidateTracker(qtrk, e, "clear results")) {
349         qtrk->ClearResults();
350     }
351 }
```

8.2.3.2 CDLL_EXPORT QueuedTracker* DLL_CALLCONV qtrk_create (QTrkSettings * settings, LStrHandle warnings, ErrorCluster * e)

```
221 {
222     QueuedTracker* tracker = 0;
223     try {
224         QTrkComputedConfig cc(*settings);
225         cc.WriteToLog();
226
227         tracker = CreateQueuedTracker(cc);
228
229         std::string w = tracker->GetWarnings();
230         if (!w.empty()) SetLVString(warnings, w.c_str());
231
232         trackerListMutex.lock();
233         trackerList.push_back(tracker);
234         trackerListMutex.unlock();
235     } catch(const std::runtime_error &exc) {
236         FillErrorCluster(kAppErrorBase, exc.what(), e );
237     }
238     return tracker;
239 }
```

8.2.3.3 CDLL_EXPORT void DLL_CALLCONV qtrk_destroy (QueuedTracker * qtrk, ErrorCluster * error)

```
242 {
243     trackerListMutex.lock();
244
245     auto pos = std::find(trackerList.begin(), trackerList.end(), qtrk);
246     if (pos == trackerList.end()) {
247         ArgumentErrorMsg(error, SPrintf( "Trying to call qtrk_destroy with invalid
qtrk pointer %p", qtrk));
248         qtrk = 0;
249     }
250     else
251         trackerList.erase(pos);
252
253     trackerListMutex.unlock();
254
255     if(qtrk) delete qtrk;
256 }
```

8.2.3.4 CDLL_EXPORT void DLL_CALLCONV qtrk_dump_memleaks()

```
501 {
502 #ifdef USE_MEMDBG
503     _CrtDumpMemoryLeaks();
504 #endif
505 }
```

8.2.3.5 CDLL_EXPORT void DLL_CALLCONV qtrk_flush(QueuedTracker * qtrk, ErrorCluster * e)

```
397 {
398     if (ValidateTracker(qtrk, e, "flush")) {
399         qtrk->Flush();
400     }
401 }
```

8.2.3.6 CDLL_EXPORT void DLL_CALLCONV qtrk_generate_image_from_lut(LVArray2D< float > ** image, LVArray2D< float > ** lut, float * LUTradii, vector2f * position, float z, float M, float sigma_noise)

```
488 {
489     ImageData img((*image)->elem, (*image)->dimSizes[1], (*image)->dimSizes[0]);
490     ImageData zlut((*lut)->elem, (*lut)->dimSizes[1], (*lut)->dimSizes[0]);
491
492     vector3f pos (position->x, position->y, z);
493     GenerateImageFromLUT(&img, &zlut, LUTradii[0], LUTradii[1], pos);
494     //img.normalize();
495     if(sigma_noise>0)
496         ApplyGaussianNoise(img, sigma_noise);
497 }
```

8.2.3.7 CDLL_EXPORT void qtrk_get_profile_report(QueuedTracker * qtrk, LStrHandle str)

```
508 {
509     SetLVString(str, qtrk->GetProfileReport().c_str());
510 }
```

8.2.3.8 CDLL_EXPORT int DLL_CALLCONV qtrk_get_results(QueuedTracker * qtrk, LocalizationResult * results, int maxResults, int sortByID, ErrorCluster * e)

```
404 {
405     if (ValidateTracker(qtrk, e, "get_results")) {
406         int resultCount = qtrk->FetchResults(results, maxResults);
407
408         if (sortByID) {
409             std::sort(results, results+resultCount, [] (decltype(*results) a, decltype(*results) b) { return
a.job.frame < b.job.frame; } );
410         }
411
412         return resultCount;
413     }
414     return 0;
415 }
```

8.2.3.9 CDLL_EXPORT void DLL_CALLCONV qtrk_get_ZLUT (QueuedTracker * tracker, LVArray3D< float > ** pzlut, ErrorCluster * e)

```

139 {
140     if (ValidateTracker(tracker, e, "get ZLUT")) {
141         int dims[3];
142
143         tracker->GetRadialZLUTSize(dims[0], dims[1], dims[2]);
144         if(dims[0]*dims[1]*dims[2]>0) {
145             ResizeLVArray3D(pzlut, dims[0], dims[1], dims[2]);
146             tracker->GetRadialZLUT( (*pzlut)->elem );
147         }
148     }
149 }
```

8.2.3.10 CDLL_EXPORT int DLL_CALLCONV qtrk_idle (QueuedTracker * qtrk, ErrorCluster * e)

```

444 {
445     if (ValidateTracker(qtrk, e, "is_idle"))
446         return qtrk->IsIdle() ? 1 : 0;
447     return 0;
448 }
```

8.2.3.11 CDLL_EXPORT void DLL_CALLCONV qtrk_queue_array (QueuedTracker * qtrk, ErrorCluster * error, LVArray2D< uchar > ** data, uint pdt, const LocalizationJob * jobInfo)

```

306 {
307     uint pitch;
308
309     if (pdt == QTrkFloat)
310         pitch = sizeof(float);
311     else if(pdt == QTrkU16)
312         pitch = 2;
313     else pitch = 1;
314
315     if (!CheckImageInput(qtrk, data, error))
316         return;
317
318     pitch *= (*data)->dimSizes[1]; // LVArray2D<uchar> type works for ushort and float as well
319 //     dbgprintf("zlutindex: %d, zlutplane: %d\n", zlutIndex,zlutPlane);
320     qtrk_queue_pitchedmem(qtrk, (*data)->elem, pitch, pdt, jobInfo);
321 }
```

8.2.3.12 CDLL_EXPORT void DLL_CALLCONV qtrk_queue_float (QueuedTracker * qtrk, ErrorCluster * error, LVArray2D< float > ** data, const LocalizationJob * jobInfo)

```

292 {
293     if (CheckImageInput(qtrk, data, error))
294     {
295         qtrk->ScheduleLocalization( (uchar*) (*data)->elem, sizeof(float)*(*data)
296         ->dimSizes[1], QTrkFloat, jobInfo);
297     }

```

8.2.3.13 CDLL_EXPORT uint DLL_CALLCONV qtrk_queue_frame (QueuedTracker * qtrk, uchar * image, int pitch, int w, int h, uint pdt, ROIPosition * pos, int numROI, const LocalizationJob * pJobInfo, QueueFrameFlags flags, ErrorCluster * e)

```

337 {
338     LocalizationJob jobInfo = *pJobInfo;
339     int nQueued;
340     if ( (nQueued=qtrk->ScheduleFrame(image, pitch, w,h, pos, numROI,
341     QTRK_PixelDataType)pdt, &jobInfo) != numROI) {
341         ArgumentErrorMsg(e, SPrintf( "Not all ROIs (%d out of %d) were queued. Check
342         image borders vs ROIs.", nQueued, numROI));
343     }
344     return jobInfo.timestamp;
345 }
```

8.2.3.14 CDLL_EXPORT void DLL_CALLCONV qtrk_queue_pitchedmem (QueuedTracker * *qtrk*, uchar * *data*, int *pitch*, uint *pdt*, const LocalizationJob * *jobInfo*)

```
301 {
302     qtrk->ScheduleLocalization(data, pitch, (
303         QTRK_PixelDataType)pdt, jobInfo);
303 }
```

8.2.3.15 CDLL_EXPORT void DLL_CALLCONV qtrk_queue_u16 (QueuedTracker * *qtrk*, ErrorCluster * *error*, LVArray2D< ushort > ** *data*, const LocalizationJob * *jobInfo*)

```
272 {
273     if (CheckImageInput(qtrk, data, error))
274     {
275         qtrk->ScheduleLocalization( (uchar*)(*data)->elem, sizeof(
276             ushort)*(*(data)->dimSizes[1], QTrkU16, jobInfo);
277     }
277 }
```

8.2.3.16 CDLL_EXPORT void DLL_CALLCONV qtrk_queue_u8 (QueuedTracker * *qtrk*, ErrorCluster * *error*, LVArray2D< uchar > ** *data*, const LocalizationJob * *jobInfo*)

```
280 {
281     if (CheckImageInput(qtrk, data, error))
282     {
283 #ifdef _DEBUG
284         //dbgprintf("Job: 8bit image, frame %d, bead %d\n", jobInfo->frame, jobInfo->zlutIndex);
285 #endif
286         qtrk->ScheduleLocalization( (*data)->elem, sizeof(
287             uchar)*(*(data)->dimSizes[1], QTrkU8, jobInfo);
288     }
289 }
```

8.2.3.17 CDLL_EXPORT uint qtrk_read_timestamp (uchar * *image*, int *w*, int *h*)

```
324 {
325     if (w*h<4) return 0;
326
327     uint ts;
328     uchar *timestamp = (uchar*)&ts;
329     // Assume little endian only
330     for (int i=0;i<4;i++)
331         timestamp[i] = image[i];
332     return ts;
333 }
```

8.2.3.18 CDLL_EXPORT int DLL_CALLCONV qtrk_resultcount (QueuedTracker * *qtrk*, ErrorCluster * *e*)

```
389 {
390     if (ValidateTracker(qtrk, e, "resultcount")) {
391         return qtrk->GetResultCount();
392     }
393     return 0;
394 }
```

8.2.3.19 CDLL_EXPORT void DLL_CALLCONV qtrk_set_ZLUT(QueuedTracker * tracker, LVArray3D< float > * pZlut, LVArray< float > ** zcmpWindow, int normalize, ErrorCluster * e)

```

101 {
102     LVArray3D<float>* zlut = *pZlut;
103
104     if (ValidateTracker(tracker, e, "set ZLUT")) {
105
106         int numLUTs = zlut->dimSizes[0];
107         int planes = zlut->dimSizes[1];
108         int res = zlut->dimSizes[2];
109
110         dbgprintf("Setting ZLUT size: %d beads, %d planes, %d radialsteps\n", numLUTs, planes, res
111     );
112
113         float* zcmp = 0;
114         if (zcmpWindow && (*zcmpWindow)->dimSize > 0) {
115             if ( (*zcmpWindow)->dimSize != res)
116                 ArgumentErrorMsg(e, Sprintf("Z Compare window should have the same
resolution as the ZLUT (%d elements)", res));
117             else
118                 zcmp = (*zcmpWindow)->elem;
119
120         if (numLUTs * planes * res == 0) {
121             tracker->SetRadialZLUT(0, 0, 0);
122         } else {
123             if (res != tracker->cfg.zlut_radialsteps)
124                 ArgumentErrorMsg(e, Sprintf("set_ZLUT: 3rd dimension should have
size of zlut_radialsteps (%d instead of %d)", tracker->cfg.zlut_radialsteps, res));
125             else {
126
127                 if (normalize) {
128                     NormalizeZLUT( zlut->elem, numLUTs, planes, res );
129                 }
130
131                 tracker->SetRadialZLUT(zlut->elem, numLUTs, planes);
132                 tracker->SetRadialWeights(zcmp);
133             }
134         }
135     }
136 }
```

8.2.3.20 CDLL_EXPORT int DLL_CALLCONV qtrkcuda_device_count(ErrorCluster * e)

```
684 { return 0; }
```

8.2.3.21 CDLL_EXPORT void DLL_CALLCONV qtrkcuda_set_device_list(LVArray< int > ** devices)

8.3 API - C

API functions available to a C or .NET program.

Functions

- **CDLL_EXPORT** QueuedTracker ***DLL_CALLCONV** QTrkCreateInstance (QTrkSettings *cfg)
Create a QTrk instance and return a pointer to it.
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkFreeInstance (QueuedTracker *qtrk)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkSetLocalizationMode (QueuedTracker *qtrk, LocMode_t locType)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkScheduleLocalization (QueuedTracker *qtrk, void *data, int pitch, QTRK_PixelDataType pdt, const LocalizationJob *jobInfo)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkClearResults (QueuedTracker *qtrk)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkFlush (QueuedTracker *qtrk)
- **CDLL_EXPORT** int **DLL_CALLCONV** QTrkScheduleFrame (QueuedTracker *qtrk, void *imgptr, int pitch, int width, int height, ROIPosition *positions, int numROI, QTRK_PixelDataType pdt, const LocalizationJob *jobInfo)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkSetRadialZLUT (QueuedTracker *qtrk, float *data, int count, int planes, float *zcmp=0)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkGetRadialZLUT (QueuedTracker *qtrk, float *dst)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkGetRadialZLUTSize (QueuedTracker *qtrk, int *count, int *planes, int *radialsteps)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkBuildLUT (QueuedTracker *qtrk, void *data, int pitch, QTRK_PixelDataType pdt, bool imageLUT, int plane)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkFinalizeLUT (QueuedTracker *qtrk)
- **CDLL_EXPORT** int **DLL_CALLCONV** QTrkGetResultCount (QueuedTracker *qtrk)
- **CDLL_EXPORT** int **DLL_CALLCONV** QTrkFetchResults (QueuedTracker *qtrk, LocalizationResult *results, int maxResults)
- **CDLL_EXPORT** int **DLL_CALLCONV** QTrkGetQueueLength (QueuedTracker *qtrk, int *maxQueueLen)
- **CDLL_EXPORT** bool **DLL_CALLCONV** QTrkIsIdle (QueuedTracker *qtrk)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkGetProfileReport (QueuedTracker *qtrk, char *dst, int maxStrLen)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkGetWarnings (QueuedTracker *qtrk, char *dst, int maxStrLen)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkGetComputedConfig (QueuedTracker *qtrk, QTrkComputedConfig *cfg)

8.3.1 Detailed Description

API functions available to a C or .NET program.

These DLLs are compiled by the *cputrack* and *cudatrack* projects.

8.3.2 Function Documentation

8.3.2.1 **CDLL_EXPORT** void **DLL_CALLCONV** QTrkBuildLUT (QueuedTracker * qtrk, void * data, int pitch, QTRK_PixelDataType pdt, bool imageLUT, int plane)

```
65 {
66     qtrk->BuildLUT(data, pitch, pdt, imageLUT, plane);
67 }
```

8.3.2.2 CDLL_EXPORT void DLL_CALLCONV QTrkClearResults (QueuedTracker * qtrk)

```
31 {
32     qtrk->ClearResults();
33 }
```

8.3.2.3 CDLL_EXPORT QueuedTracker* DLL_CALLCONV QTrkCreateInstance (QTrkSettings * cfg)

Create a QTrk instance and return a pointer to it.

Parameters

in	<i>cfg</i>	Pointer to the structure with the desired tracking settings.
----	------------	--

Returns

Pointer to the created QTrk instance.

```
6 {
7     return CreateQueuedTracker(*cfg);
8 }
```

8.3.2.4 CDLL_EXPORT int DLL_CALLCONV QTrkFetchResults (QueuedTracker * qtrk, LocalizationResult * results, int maxResults)

```
81 {
82     return qtrk->FetchResults(results, maxResults);
83 }
```

8.3.2.5 CDLL_EXPORT void DLL_CALLCONV QTrkFinalizeLUT (QueuedTracker * qtrk)

```
70 {
71     qtrk->FinalizeLUT();
72 }
```

8.3.2.6 CDLL_EXPORT void DLL_CALLCONV QTrkFlush (QueuedTracker * qtrk)

```
36 {
37     qtrk->Flush();
38 }
```

8.3.2.7 CDLL_EXPORT void DLL_CALLCONV QTrkFreeInstance (QueuedTracker * qtrk)

```
11 {
12     delete qtrk;
13 }
```

8.3.2.8 CDLL_EXPORT void DLL_CALLCONV QTrkGetComputedConfig (QueuedTracker * *qtrk*, QTrkComputedConfig * *cfg*)

```
108 {  
109     *cfg = qtrk->cfg;  
110 }
```

8.3.2.9 CDLL_EXPORT void DLL_CALLCONV QTrkGetProfileReport (QueuedTracker * *qtrk*, char * *dst*, int *maxStrLen*)

```
97 {  
98     strncpy(dst, qtrk->GetProfileReport().c_str(), maxStrLen);  
99 }
```

8.3.2.10 CDLL_EXPORT int DLL_CALLCONV QTrkGetQueueLength (QueuedTracker * *qtrk*, int * *maxQueueLen*)

```
87 {  
88     return qtrk->GetQueueLength(maxQueueLen);  
89 }
```

8.3.2.11 CDLL_EXPORT void DLL_CALLCONV QTrkGetRadialZLUT (QueuedTracker * *qtrk*, float * *dst*)

```
54 {  
55     qtrk->GetRadialZLUT(dst);  
56 }
```

8.3.2.12 CDLL_EXPORT void DLL_CALLCONV QTrkGetRadialZLUTSize (QueuedTracker * *qtrk*, int * *count*, int * *planes*, int * *radialsteps*)

```
59 {  
60     qtrk->GetRadialZLUTSize(*count, *planes, *radialsteps);  
61 }
```

8.3.2.13 CDLL_EXPORT int DLL_CALLCONV QTrkGetResultCount (QueuedTracker * *qtrk*)

```
76 {  
77     return qtrk->GetResultCount();  
78 }
```

8.3.2.14 CDLL_EXPORT void DLL_CALLCONV QTrkGetWarnings (QueuedTracker * *qtrk*, char * *dst*, int *maxStrLen*)

```
102 {  
103     strncpy(dst, qtrk->GetWarnings().c_str(), maxStrLen);  
104 }
```

8.3.2.15 CDLL_EXPORT bool DLL_CALLCONV QTrkIsIdle (QueuedTracker * qtrk)

```
92 {  
93     return qtrk->IsIdle();  
94 }
```

**8.3.2.16 CDLL_EXPORT int DLL_CALLCONV QTrkScheduleFrame (QueuedTracker * qtrk, void * imgptr, int pitch,
int width, int height, ROIPosition * positions, int numROI, QTRK_PixelDataType pdt, const LocalizationJob
* jobInfo)**

```
42 {  
43     return qtrk->ScheduleFrame(imgptr, pitch, width, height, positions, numROI, pdt, jobInfo);  
44 }
```

**8.3.2.17 CDLL_EXPORT void DLL_CALLCONV QTrkScheduleLocalization (QueuedTracker * qtrk, void * data, int
pitch, QTRK_PixelDataType pdt, const LocalizationJob * jobInfo)**

```
26 {  
27     qtrk->ScheduleLocalization(data, pitch, pdt, jobInfo);  
28 }
```

**8.3.2.18 CDLL_EXPORT void DLL_CALLCONV QTrkSetLocalizationMode (QueuedTracker * qtrk, LocMode_t
locType)**

```
18 {  
19     qtrk->SetLocalizationMode(locType);  
20 }
```

**8.3.2.19 CDLL_EXPORT void DLL_CALLCONV QTrkSetRadialZLUT (QueuedTracker * qtrk, float * data, int count,
int planes, float * zcmp = 0)**

```
49 {  
50     qtrk->SetRadialZLUT(data, count, planes, zcmp);  
51 }
```

Chapter 9

Namespace Documentation

9.1 BeadFinder Namespace Reference

Classes

- struct [Config](#)
- struct [Position](#)

Functions

- std::vector< [Position](#) > [Find](#) ([ImageData](#) *img, float *sample, [Config](#) *cfg)
- std::vector< [Position](#) > [Find](#) (uint8_t *img, int pitch, int w, int h, int smpCornerX, int smpCornerY, [Config](#) *cfg)

9.1.1 Function Documentation

9.1.1.1 std::vector< [Position](#) > [BeadFinder::Find](#) ([ImageData](#) * *img*, float * *sample*, [Config](#) * *cfg*)

```
169 {  
170     typedef std::complex<float> pixelc_t;  
171  
172     // Compute nearest power of two dimension  
173     int w=NextPowerOf2(img->w);  
174     int h=NextPowerOf2(img->h);  
175  
176     // Convert to std::complex and copy the image in there (subtract mean first)  
177     float mean = img->mean();  
178     pixelc_t* cimg = new pixelc_t[w*h];  
179     std::fill(cimg, cimg+w*h, pixelc_t());  
180     for (int y=0;y->h;y++) {  
181         for (int x=0;x->w;x++) {  
182             cimg[y*w+x] = img->data[y->w+x]-mean;  
183         }  
184     }  
185  
186     // Inplace 2D fft  
187     FFT2D(cimg,w,h,false);  
188  
189     // Create an image to put the sample in with size [w,h] as well  
190     pixelc_t* smpimg = new pixelc_t[w*h];  
191     std::fill(smpimg, smpimg+w*h, pixelc_t());  
192     float smpmean = ImageData(sample,cfg->roi,cfg->roi).mean();  
193     for (int y=0;y->roi;y++)  
194         for (int x=0;x->roi;x++)  
195             smpimg[ ((y-cfg->roi) * w + ( (x-cfg->roi) & (w-1) ) ] = sample[cfg->  
roi*y+x]-smpmean;
```

```

196 // ComplexToJPEGFile("smpimg.jpg", smpimg, w,h);
197 FFT2D(smpimg,w,h,false);
198
199 for (int i=0;i<w*h;i++)
200     cimg[i]*=smpimg[i];
201     FFT2D(cimg,w,h,true);
202
203     float maxVal = 0.0f;
204     for (int i=0;i<w*h;i++)
205         maxVal=std::max(maxVal, cimg[i].real());
206
207     std::vector<Position> pts;
208     for (int y=0;y<img->h;y++) {
209         for (int x=0;x<img->w;x++) {
210             if (cimg[y*w+x].real()>maxVal*cfg->similarity)
211                 pts.push_back (SearchArea(cimg, w, h, x,y, cfg->
212 MinPixelDistance()));
213         }
214     }
215 //ComplexToJPEGFile("result.jpg", cimg, w,h);
216 delete[] smpimg;
217 delete[] cimg;
218
219 RecenterAndFilter(img, pts, cfg);
220
221 return pts;
222 }

```

9.1.1.2 `std::vector< Position > BeadFinder::Find (uint8_t * img, int pitch, int w, int h, int smpCornerX, int smpCornerY, BeadFinder::Config * cfg)`

```

225 {
226     ImageData fimg = ImageData::alloc(w,h);
227
228     for (int y=0;y<h;y++)
229         for (int x=0;x<w;x++)
230             fimg.at(x,y) = img[y*pitch+x];
231
232     float *fsmp = new float[cfg->roi*cfg->roi];
233     for (int y=0;y<cfg->roi;y++) {
234         for (int x=0;x<cfg->roi;x++) {
235             fsmp [y*cfg->roi+x] = fimg [ (y+smpCornerY) * h + x + smpCornerX ];
236         }
237     }
238
239     std::vector<Position> beads = Find(&fimg, fsmp, cfg);
240
241     delete[] fsmp;
242     fimg.free();
243     return beads;
244 }

```

9.2 kissfft_utils Namespace Reference

Classes

- struct `traits`

9.3 qtrk Namespace Reference

Classes

- class `hash_map`
- class `hash_set`

9.4 sfft Namespace Reference

Classes

- struct **complex**

Functions

- template<typename T >
SFFT_BOTH void **swap** (T &a, T &b)
- template<typename T , int sign>
SFFT_BOTH void **fft** (size_t N, std::complex< T > *zs, **complex**< T > *twiddles)
- template<typename T >
std::vector< **complex**< T > > **fill_twiddles** (int N)
- template<typename T >
SFFT_BOTH void **fft_forward** (size_t N, **complex**< T > *zs, **complex**< T > *twiddles)
- template<typename T >
SFFT_BOTH void **fft_inverse** (size_t N, **complex**< T > *zs, **complex**< T > *twiddles)

9.4.1 Function Documentation

9.4.1.1 template<typename T , int sign> **SFFT_BOTH** void **sfft::fft** (size_t N, std::complex< T > * zs, **complex**< T > * twiddles)

```

43
44     unsigned int j=0;
45     if (sign < 0)
46         twiddles += N; // forward FFT twiddles are located after inverse twiddles
47 // Warning about signed vs unsigned comparison
48     for(unsigned int i=0; i<N-1; ++i) {
49         if (i < j)
50             swap(zs[i], zs[j]);
51         int m=N/2;
52         j^=m;
53         while ((j & m) == 0) { m/=2; j^=m; }
54     }
55     for(unsigned int j=1; j<N; j*=2)
56         for(unsigned int m=0; m<j; ++m) {
57             //T t = pi * sign * m / j;
58             //dbgprintf("fac: m=%d. j=%d. k=%d\n", m, j, N*m/(j*2));
59             //complex<T> wcmp = complex<T>(cos(t),sin(t));
60             complex<T> w = twiddles[N*m/(j*2)];
61             for(unsigned int i = m; i<N; i+=2*j) {
62                 complex<T> zi = zs[i], t = w * zs[i + j];
63                 zs[i] = zi+t;
64                 zs[i+j] = zi-t;
65             }
66         }
67 }
```

9.4.1.2 template<typename T > **SFFT_BOTH** void **sfft::fft_forward** (size_t N, **complex**< T > * zs, **complex**< T > * twiddles)

```

87
88     fft<T, -1>(N,zs,twiddles);
89 }
```

9.4.1.3 template<typename T > **SFFT_BOTH** void sfft::fft_inverse (size_t *N*, complex< T > * *zs*, complex< T > * *twiddles*)

```
91
92     fft<T, 1>(N,zs,twiddles);
93 }
```

9.4.1.4 template<typename T > std::vector< complex<T> > sfft::fill_twiddles (int *N*)

```
71
72     const T pi = 3.14159265359;
73     std::vector< complex<T> > twiddles(N*2);
74     for(int i=0;i<N;i++) {
75         T t = pi * 1 * i * 2 / N;
76         twiddles[i] = complex<T>(cos(t),sin(t));
77     }
78     for(int i=0;i<N;i++) {
79         T t = pi * -1 * i * 2 / N;
80         twiddles[i+N] = complex<T>(cos(t),sin(t));
81     }
82     return twiddles;
83 }
```

9.4.1.5 template<typename T > **SFFT_BOTH** void sfft::swap (T & *a*, T & *b*)

```
35
36     T tmp(a);
37     a = b;
38     b = tmp;
39 }
```

Chapter 10

Class Documentation

10.1 Atomic< T > Class Template Reference

```
#include <threads.h>
```

Public Member Functions

- `Atomic (const T &o=T())`
- `operator T () const`
- `Atomic & operator=(const T &x)`
- `void set (const T &x)`
- `T get () const`

Private Attributes

- `Threads::Mutex m`
- `T data`

10.1.1 Constructor & Destructor Documentation

10.1.1.1 template<typename T> Atomic< T >::Atomic (const T & o = T ()) [inline]

```
160 : data(o) {} // no need for locking: the object is not allowed to be used before the constructor is  
done anyway
```

10.1.2 Member Function Documentation

10.1.2.1 template<typename T> T Atomic< T >::get () const [inline]

```
168 {  
169     m.lock();  
170     T x=data;  
171     m.unlock();  
172     return x;  
173 }
```

10.1.2.2 template<typename T> Atomic< T >::operator T() const [inline]

```
161 { return get(); };
```

10.1.2.3 template<typename T> Atomic& Atomic< T >::operator=(const T & x) [inline]

```
162 { set(x); return *this; }
```

10.1.2.4 template<typename T> void Atomic< T >::set(const T & x) [inline]

```
163             {
164         m.lock();
165         data=x;
166         m.unlock();
167     }
```

10.1.3 Member Data Documentation

10.1.3.1 template<typename T> T Atomic< T >::data [private]

10.1.3.2 template<typename T> Threads::Mutex Atomic< T >::m [mutable], [private]

The documentation for this class was generated from the following file:

- cputrack/[threads.h](#)

10.2 BaseKernelParams Struct Reference

```
#include <QueuedCUDATracker.h>
```

Public Attributes

- int [njobs](#)
- [LocalizationParams](#) * [locParams](#)
- float * [imgmeans](#)
- [cudalImageListf](#) [images](#)

10.2.1 Member Data Documentation

10.2.1.1 [cudalImageListf](#) BaseKernelParams::[images](#)

10.2.1.2 float* BaseKernelParams::[imgmeans](#)

10.2.1.3 [LocalizationParams](#)* BaseKernelParams::[locParams](#)

10.2.1.4 int BaseKernelParams::[njobs](#)

The documentation for this struct was generated from the following file:

- cudatrack/[QueuedCUDATracker.h](#)

10.3 BenchmarkLUT Class Reference

```
#include <BenchmarkLUT.h>
```

Public Member Functions

- `BenchmarkLUT ()`
- `BenchmarkLUT (ImageData *lut)`
- `BenchmarkLUT (const char *file)`
- `void Load (ImageData *lut)`
- `void Load (const char *file)`
- `void GenerateLUT (ImageData *lut)`
- `void GenerateSample (ImageData *image, vector3f pos, float minRadius, float maxRadius)`

Static Public Member Functions

- `static void CleanupLUT (ImageData &lut)`

Public Attributes

- `std::vector< float > normprof`
- `float max_a`
- `float max_b`
- `float max_c`
- `int lut_w`
- `int lut_h`

10.3.1 Constructor & Destructor Documentation

10.3.1.1 BenchmarkLUT::BenchmarkLUT() [inline]

```
15 { lut_w=lut_h=0; }
```

10.3.1.2 BenchmarkLUT::BenchmarkLUT(ImageData * lut)

```
11 {
12     Load(lut);
13 }
```

10.3.1.3 BenchmarkLUT::BenchmarkLUT(const char * file)

```
6 {
7     Load(file);
8 }
```

10.3.2 Member Function Documentation

10.3.2.1 void BenchmarkLUT::CleanupLUT (`ImageData & lut`) [static]

```
102 {
103     BenchmarkLUT bm(&lut);
104     bm.GenerateLUT(&lut);
105 }
```

10.3.2.2 void BenchmarkLUT::GenerateLUT (`ImageData * lut`)

```
71 {
72     float M = lut->w / (float)lut_w;
73
74     for (int y=0;y<lut->h;y++) {
75         for (int x=0;x<lut->w;x++) {
76             float maxline=M*(max_a*y*y+max_b*y+max_c)/max_c;
77             lut->at(x,y) = Interpolate1D(normprof, x/maxline);
78         }
79     }
80 }
```

10.3.2.3 void BenchmarkLUT::GenerateSample (`ImageData * image, vector3f pos, float minRadius, float maxRadius`)

```
83 {
84     float radialDensity=lut_w / (maxRadius-minRadius);
85
86     if(pos.z<0.0f) pos.z=0.0f;
87     if(pos.z>lut_h-1) pos.z=lut_h-1;
88
89     for (int y=0;y<image->h;y++)
90         for (int x=0;x<image->w;x++)
91     {
92         float dx=x-pos.x;
93         float dy=y-pos.y;
94         float r = (sqrt(dx*dx+dy*dy)-minRadius)*radialDensity; // r in original radial bin pixels
95         float maxline=(max_a*pos.z*pos.z+max_b*pos.z+max_c) /
max_c;
96         float profpos = r / maxline;
97         image->at(x,y) = Interpolate1D(normprof, profpos);
98     }
99 }
```

10.3.2.4 void BenchmarkLUT::Load (`ImageData * lut`)

```
23 {
24     std::vector<float> max_x,max_y;
25
26     lut_w = lut->w;
27     lut_h = lut->h;
28
29     for (int i=lut->h/5;i<lut->h*3/4;i++) {
30         int maxpos = 0;
31         float maxval = lut->at(0, i);
32         for (int x=1;x<lut->w;x++)
33             if (maxval < lut->at(x,i)) {
34                 maxval = lut->at(x,i);
35                 maxpos=x;
36             }
37             max_x.push_back(maxpos);
38             max_y.push_back(i);
39     }
40
41     LsqSqQuadFit<float> fit(max_x.size(), &max_y[0],&max_x[0]);
42
43     max_a = fit.a;
44     max_b = fit.b;
45     max_c = fit.c; //normalized
```

```

46     std::vector<int> numsmpl(lut->w);
47     normprof.resize(lut->w);
48     for (int r=0;r<lut->w;r++) {
49         float sum=0.0f;
50         int nsmp=0;
51         for (int y=0;y<lut->h;y++) {
52             float x=r*fit.compute(y)/fit.c;
53             bool outside=false;
54             float v = lut->interpolate(x,y,&outside);
55             if (!outside) {
56                 sum+=v;
57                 nsmp++;
58             }
59         }
60         numsmpl[r]=nsmp;
61         normprof[r]=sum;
62     }
63     for (int i=0;i<lut->w;i++) {
64         if (numsmpl[i] == 0 && i>0) normprof[i]=normprof[i-1];
65         else normprof[i]=normprof[i]/numsmpl[i];
66     }
67 }
68 }
```

10.3.2.5 void BenchmarkLUT::Load (const char *file)

```

16 {
17     ImageData d = ReadJPEGFile(file);
18     Load(&d);
19     d.free();
20 }
```

10.3.3 Member Data Documentation

10.3.3.1 int BenchmarkLUT::lut_h

10.3.3.2 int BenchmarkLUT::lut_w

10.3.3.3 float BenchmarkLUT::max_a

10.3.3.4 float BenchmarkLUT::max_b

10.3.3.5 float BenchmarkLUT::max_c

10.3.3.6 std::vector<float> BenchmarkLUT::normprof

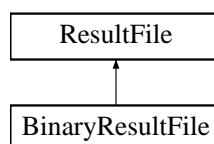
The documentation for this class was generated from the following files:

- cputrack/BenchmarkLUT.h
- cputrack/BenchmarkLUT.cpp

10.4 BinaryResultFile Class Reference

```
#include <ResultManager.h>
```

Inheritance diagram for BinaryResultFile:



Public Member Functions

- [BinaryResultFile](#) (const char *fn, bool write)
- void [LoadRow](#) (std::vector< [vector3f](#) > &pos)
- void [SaveRow](#) (std::vector< [vector3f](#) > &pos)

Protected Attributes

- FILE * f

10.4.1 Constructor & Destructor Documentation

10.4.1.1 [BinaryResultFile::BinaryResultFile](#) (const char * *fn*, bool *write*)

10.4.2 Member Function Documentation

10.4.2.1 void [BinaryResultFile::LoadRow](#) (std::vector< [vector3f](#) > & *pos*) [virtual]

Implements [ResultFile](#).

```
22 {  
23  
24 }
```

10.4.2.2 void [BinaryResultFile::SaveRow](#) (std::vector< [vector3f](#) > & *pos*) [virtual]

Implements [ResultFile](#).

```
27 {  
28 }
```

10.4.3 Member Data Documentation

10.4.3.1 FILE* [BinaryResultFile::f](#) [protected]

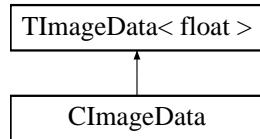
The documentation for this class was generated from the following files:

- cputrack/[ResultManager.h](#)
- cputrack/[ResultManager.cpp](#)

10.5 CImageData Class Reference

```
#include <utils.h>
```

Inheritance diagram for CImageData:



Public Member Functions

- [CImageData \(int w, int h\)](#)
- [CImageData \(const CImageData &other\)](#)
- [CImageData \(\)](#)
- [~CImageData \(\)](#)
- [CImageData \(const TImageData< float > &src\)](#)
- [CImageData & operator= \(const CImageData &src\)](#)
- [CImageData & operator= \(const TImageData< float > &src\)](#)

Additional Inherited Members

10.5.1 Constructor & Destructor Documentation

10.5.1.1 CImageData::CImageData (int w, int h) [inline]

```

117     : TImageData<float>(new float[w*h],
118     w, h)
119   {
  
```

10.5.1.2 CImageData::CImageData (const CImageData & other) [inline]

```

119     data=0; set(other);
120   }
121   {
  
```

10.5.1.3 CImageData::CImageData () [inline]

```
122 { }
```

10.5.1.4 CImageData::~CImageData () [inline]

```
123 { free(); }
```

10.5.1.5 ClImageData::ClImageData (const TImageData< float > & src) [inline]

```
124
125     data=0; set(src);
126 }
```

10.5.2 Member Function Documentation

10.5.2.1 ClImageData& ClImageData::operator= (const ClImageData & src) [inline]

```
128
129     set(src);
130     return *this;
131 }
```

10.5.2.2 ClImageData& ClImageData::operator= (const TImageData< float > & src) [inline]

```
132
133     set(src);
134     return *this;
135 }
```

The documentation for this class was generated from the following file:

- cputrack/utils.h

10.6 LsqSqQuadFit< T >::Coeff Struct Reference

```
#include <LsqQuadraticFit.h>
```

Public Member Functions

- CUDA_SUPPORTED_FUNC void abc (T &a, T &b, T &c, T &d)

Public Attributes

- T s40
- T s30
- T s20
- T s10
- T s21
- T s11
- T s01
- T s00

10.6.1 Member Function Documentation

10.6.1.1 template<typename T> CUDA_SUPPORTED_FUNC void LsqSqQuadFit< T >::Coeff::abc (T & a, T & b, T & c, T & d) [inline]

```

18
19      d = s40 * (s20 * s00 - s10 * s10) - s30 * (s30 *
20      s00 - s10 * s20) + s20 * (s30 * s10 - s20 * s20);
21
22      a = (s21*(s20 * s00 - s10 * s10) - s11*(s30 *
23      s00 - s10 * s20) + s01*(s30 * s10 - s20 * s20)) / d;
24      b = (s40*(s11 * s00 - s01 * s10) - s30*(s21 *
25      s00 - s01 * s20) + s20*(s21 * s10 - s11 * s20)) / d;
26      c = (s40*(s20 * s01 - s10 * s11) - s30*(s30 *
27      s01 - s10 * s21) + s20*(s30 * s11 - s20 * s21)) / d;
28  }
```

10.6.2 Member Data Documentation

10.6.2.1 template<typename T> T LsqSqQuadFit< T >::Coeff::s00

10.6.2.2 template<typename T> T LsqSqQuadFit< T >::Coeff::s01

10.6.2.3 template<typename T> T LsqSqQuadFit< T >::Coeff::s10

10.6.2.4 template<typename T> T LsqSqQuadFit< T >::Coeff::s11

10.6.2.5 template<typename T> T LsqSqQuadFit< T >::Coeff::s20

10.6.2.6 template<typename T> T LsqSqQuadFit< T >::Coeff::s21

10.6.2.7 template<typename T> T LsqSqQuadFit< T >::Coeff::s30

10.6.2.8 template<typename T> T LsqSqQuadFit< T >::Coeff::s40

The documentation for this struct was generated from the following file:

- cputrack/[LsqQuadraticFit.h](#)

10.7 sfft::complex< T > Struct Template Reference

```
#include <simplefft.h>
```

Public Member Functions

- `SFFT_BOTH T & imag ()`
- `SFFT_BOTH T & real ()`
- `SFFT_BOTH const T & imag () const`
- `SFFT_BOTH const T & real () const`
- `SFFT_BOTH complex ()`
- `SFFT_BOTH complex (T a, T b=0.0)`
- `SFFT_BOTH complex conjugate ()`
- `SFFT_BOTH complex operator* (const T &b) const`
- `SFFT_BOTH complex & operator*=(const T &b)`
- `SFFT_BOTH complex operator* (const complex &b) const`
- `SFFT_BOTH complex operator- (const complex &b) const`
- `SFFT_BOTH complex operator+ (const complex &b) const`
- `SFFT_BOTH complex & operator+=(const complex &b)`
- `complex (const std::complex< T > &a)`

Public Attributes

- `T x`
- `T y`

10.7.1 Constructor & Destructor Documentation

10.7.1.1 template<typename T> **SFFT_BOTH** sfft::complex< T >::complex() [inline]

```
20 : x(0.0f), y(0.0f) {}
```

10.7.1.2 template<typename T> **SFFT_BOTH** sfft::complex< T >::complex(T a, T b = 0.0) [inline]

```
21 : x(a), y(b) {}
```

10.7.1.3 template<typename T> sfft::complex< T >::complex(const std::complex< T > & a) [inline]

```
31 : x(a.real()), y(a.imag()) {}
```

10.7.2 Member Function Documentation

10.7.2.1 template<typename T> **SFFT_BOTH** complex sfft::complex< T >::conjugate() [inline]

```
22 { return complex(x, -y); }
```

10.7.2.2 template<typename T> **SFFT_BOTH** T& sfft::complex< T >::imag() [inline]

```
16 { return y; }
```

10.7.2.3 template<typename T> SFFT_BOTH const T& sfft::complex< T >::imag() const [inline]

```
18 { return y; }
```

10.7.2.4 template<typename T> SFFT_BOTH complex sfft::complex< T >::operator*(const T & b) const [inline]

```
24 { return complex(x*b, y*b); }
```

10.7.2.5 template<typename T> SFFT_BOTH complex sfft::complex< T >::operator*(const complex< T > & b) const [inline]

```
26 { return complex(x*b.x - y*b.y, x*b.y + y*b.x); }
```

10.7.2.6 template<typename T> SFFT_BOTH complex& sfft::complex< T >::operator*=(const T & b) [inline]

```
25 { x*=b; y*=b; return *this; }
```

10.7.2.7 template<typename T> SFFT_BOTH complex sfft::complex< T >::operator+(const complex< T > & b) const [inline]

```
28 { return complex(x+b.x, y+b.y); }
```

10.7.2.8 template<typename T> SFFT_BOTH complex& sfft::complex< T >::operator+=(const complex< T > & b) [inline]

```
29 { x+=b.x; y+=b.y; return *this; }
```

10.7.2.9 template<typename T> SFFT_BOTH complex sfft::complex< T >::operator-(const complex< T > & b) const [inline]

```
27 { return complex(x-b.x, y-b.y); }
```

10.7.2.10 template<typename T> SFFT_BOTH T& sfft::complex< T >::real() [inline]

```
17 { return x; }
```

10.7.2.11 template<typename T> SFFT_BOTH const T& sfft::complex< T >::real() const [inline]

```
19 { return x; }
```

10.7.3 Member Data Documentation

10.7.3.1 template<typename T> T sfft::complex< T >::x

10.7.3.2 template<typename T> T sfft::complex< T >::y

The documentation for this struct was generated from the following file:

- cudatrack/simplefft.h

10.8 ComputeMaxInterp< T, numPts > Class Template Reference

```
#include <LsqQuadraticFit.h>
```

Static Public Member Functions

- static CUDA_SUPPORTED_FUNC T max_ (T a, T b)
- static CUDA_SUPPORTED_FUNC T min_ (T a, T b)
- static CUDA_SUPPORTED_FUNC T Compute (T *data, int len, const T *weights, LsqSqQuadFit< T > *fit=0)

10.8.1 Member Function Documentation

10.8.1.1 template<typename T , int numPts = 3> static CUDA_SUPPORTED_FUNC T ComputeMaxInterp< T, numPts >::Compute (T * data, int len, const T * weights, LsqSqQuadFit< T > * fit = 0) [inline], [static]

```

147 {
148     int iMax=0;
149     T vMax=data[0];
150     for (int k=1;k<len;k++) {
151         if (data[k]>vMax) {
152             vMax = data[k];
153             iMax = k;
154         }
155     }
156     T xs[numPts];
157     int startPos = max_(iMax-numPts/2, 0);
158     int endPos = min_(iMax+(numPts-numPts/2), len);
159     int numpoints = endPos - startPos;
160
161     if (numpoints<3)
162         return iMax;
163     else {
164         for(int i=startPos;i<endPos;i++)
165             xs[i-startPos] = i-iMax;
166
167         LsqSqQuadFit<T> qfit(numpoints, xs, &data[startPos], weights);
168         if (fit) *fit = qfit;
169         //printf("iMax: %d. qfit: data[%d]=%f\n", iMax, startPos, data[startPos]);
170         //for (int k=0;k<numpoints;k++) {
171         //    printf("data[%d]=%f\n", startPos+k, data[startPos]);
172         //}
173
174         if (fabs(qfit.a)<1e-9f)
175             return (T)iMax;
176         else {
177             T interpMax = qfit.maxPos();
178             return (T)iMax + interpMax;
179         }
180     }
181 }
```

```
10.8.1.2 template<typename T , int numPts = 3> static CUDA_SUPPORTED_FUNC T ComputeMaxInterp< T, numPts
>::max_ ( T a, T b ) [inline], [static]
```

```
143 { return a>b ? a : b; }
```

```
10.8.1.3 template<typename T , int numPts = 3> static CUDA_SUPPORTED_FUNC T ComputeMaxInterp< T, numPts
>::min_ ( T a, T b ) [inline], [static]
```

```
144 { return a<b ? a : b; }
```

The documentation for this class was generated from the following file:

- cputrack/[LsqQuadraticFit.h](#)

10.9 BeadFinder::Config Struct Reference

```
#include <BeadFinder.h>
```

Public Member Functions

- float [MinPixelDistance \(\)](#)

Public Attributes

- int [roi](#)
- float [img_distance](#)
- float [similarity](#)

10.9.1 Member Function Documentation

10.9.1.1 float BeadFinder::Config::MinPixelDistance () [inline]

```
18 { return 0.5f * img_distance * roi; }
```

10.9.2 Member Data Documentation

10.9.2.1 float BeadFinder::Config::img_distance

10.9.2.2 int BeadFinder::Config::roi

10.9.2.3 float BeadFinder::Config::similarity

The documentation for this struct was generated from the following file:

- cputrack/[BeadFinder.h](#)

10.10 CPUTracker Class Reference

```
#include <cpu_tracker.h>
```

Classes

- class [FFT2D](#)
- struct [Gauss2DResult](#)

Public Types

- enum [LUTProfileMaxComputeMode](#) { [LUTProfMaxQuadraticFit](#), [LUTProfMaxSplineFit](#), [LUTProfMaxSimpleInterp](#) }

Public Member Functions

- float * [GetRadialZLUT](#) (int index)
- float & [GetPixel](#) (int x, int y)
- int [GetWidth](#) ()
- int [GetHeight](#) ()
- [CPUTracker](#) (int w, int h, int xcorwindow=128, bool [testRun](#)=false)
- [~CPUTracker](#) ()
- bool [KeepInsideBoundaries](#) (vector2f *center, float radius)
- bool [CheckBoundaries](#) (vector2f center, float radius)
- vector2f [ComputeXCorlInterpolated](#) (vector2f initial, int iterations, int profileWidth, bool &boundaryHit)
- vector2f [ComputeQI](#) (vector2f initial, int iterations, int radialSteps, int angularStepsPerQuadrant, float angStepIterationFactor, float minRadius, float maxRadius, bool &boundaryHit, float *radialWeights=0)
- [Gauss2DResult](#) [Compute2DGaussianMLE](#) (vector2f initial, int iterations, float sigma)
- scalar_t [QI_CompOffset](#) (complex_t *qi_profile, int nr, int axisForDebug)
- scalar_t [QuadrantAlign_CompOffset](#) (complex_t *profile, complex_t *zlut_prof_fft, int nr, int axisForDebug)
- float [ComputeAsymmetry](#) (vector2f center, int radialSteps, int angularSteps, float minRadius, float maxRadius, float *dstAngProf=0)
- template<typename TPixel >
void [SetImage](#) (TPixel *srclImage, uint srcPitch)
- void [SetImage16Bit](#) (ushort *srclImage, uint srcPitch)
- void [SetImage8Bit](#) (uchar *srclImage, uint srcPitch)
- void [SetImageFloat](#) (float *srclImage)
- void [SaveImage](#) (const char *filename)
- vector2f [ComputeMeanAndCOM](#) (float bgcorrection=0.0f)
- void [ComputeRadialProfile](#) (float *dst, int radialSteps, int angularSteps, float minradius, float maxradius, vector2f center, bool crp, bool *boundaryHit=0, bool normalize=true)
- void [ComputeQuadrantProfile](#) (scalar_t *dst, int radialSteps, int angularSteps, int quadrant, float minRadius, float maxRadius, vector2f center, float *radialWeights=0)
- float [ComputeZ](#) (vector2f center, int angularSteps, int zlutIndex, bool *boundaryHit=0, float *profile=0, float *cmpprof=0, bool normalizeProfile=true)
- void [FourierTransform2D](#) ()
- void [FourierRadialProfile](#) (float *dst, int radialSteps, int angularSteps, float minradius, float maxradius)
- void [Normalize](#) (float *image=0)
- void [SetRadialZLUT](#) (float *data, int planes, int res, int num_zluts, float minradius, float maxradius, bool copyMemory, bool useCorrelation)

- void `SetRadialWeights` (float *w)
- void `CalculateErrorCurve` (double *errorcurve_dest, float *profile, float *zlut_sel)
- void `CalculateInterpolatedZLUTProfile` (float *profile_dest, float z, int zlutIndex)
- float `CalculateErrorFlag` (double *prof1, double *prof2)
- float `LUTProfileCompare` (float *profile, int zlutIndex, float *cmpProf, `LUTProfileMaxComputeMode` maxPosMethod, float *lsqfittedcurve=0, int *maxPos=0, int frameNum=0)
- float `LUTProfileCompareAdjustedWeights` (float *rprof, int zlutIndex, float z_estim)
- float * `GetDebugImage` ()
- void `ApplyOffsetGain` (float *offset, float *gain, float offsetFactor, float gainFactor)
- void `AllocateQIFFTs` (int nsteps)
- `vector3f QuadrantAlign` (`vector3f` initial, int beadIndex, int angularStepsPerQuadrant, bool &boundaryHit)

Public Attributes

- int `width`
- int `height`
- int `xcorw`
- int `trackerID`
- float * `srclImage`
- float * `debugImage`
- float `mean`
- float `stdev`
- std::vector< `vector2f` > `radialDirs`
- float * `zluts`
- bool `zlut_memoryOwner`
- int `zlut_planes`
- int `zlut_res`
- int `zlut_count`
- float `zlut_minradius`
- float `zlut_maxradius`
- bool `zlut_useCorrelation`
- std::vector< float > `zlut_radialweights`
- `kissfft< scalar_t > *` `qa_fft_forward`
- `kissfft< scalar_t > *` `qa_fft_backward`
- bool `testRun`
- `XCor1DBuffer` * `xcorBuffer`
- std::vector< `vector2f` > `quadrantDirs`
- int `qi_radialsteps`
- `kissfft< scalar_t > *` `qi_fft_forward`
- `kissfft< scalar_t > *` `qi_fft_backward`
- `FFT2D` * `fft2d`

10.10.1 Member Enumeration Documentation

10.10.1.1 enum CPUTracker::LUTProfileMaxComputeMode

Enumerator

- `LUTProfMaxQuadraticFit`*
- `LUTProfMaxSplineFit`*
- `LUTProfMaxSimpleInterp`*

```
110 { LUTProfMaxQuadraticFit, LUTProfMaxSplineFit,
      LUTProfMaxSimpleInterp };
```

10.10.2 Constructor & Destructor Documentation

10.10.2.1 CPUTracker::CPUTracker (int *w*, int *h*, int *xcorwindow* = 128, bool *testRun* = false)

```

43 {
44     width = w;
45     height = h;
46     trackerID = 0;
47
48     xcorBuffer = 0;
49
50     mean=0.0f;
51     srcImage = new float [w*h];
52     debugImage = new float [w*h];
53     std::fill(srcImage, srcImage+w*h, 0.0f);
54     std::fill(debugImage, debugImage+w*h, 0.0f);
55
56     zluts = 0;
57     zlut_planes = zlut_res = zlut_count = 0;
58     zlut_minradius = zlut_maxradius = 0.0f;
59     xcorw = xcorwindow;
60     qa_fft_forward = qa_fft_backward = 0;
61
62     qi_radialsteps = 0;
63     qi_fft_forward = qi_fft_backward = 0;
64
65     fft2d=0;
66
67     testRun = testMode;
68 }
```

10.10.2.2 CPUTracker::~CPUTracker ()

```

71 {
72     delete[] srcImage;
73     delete[] debugImage;
74     if (zluts && zlut_memoryOwner)
75         delete[] zluts;
76
77     if (xcorBuffer)
78         delete xcorBuffer;
79
80     if (qi_fft_forward) {
81         delete qi_fft_forward;
82         delete qi_fft_backward;
83     }
84
85     if (qa_fft_backward) {
86         delete qa_fft_backward;
87         delete qa_fft_forward;
88     }
89
90     if (fft2d)
91         delete fft2d;
92 }
```

10.10.3 Member Function Documentation

10.10.3.1 void CPUTracker::AllocateQIFFTs (int *nsteps*)

```

276 {
277     if(!qi_fft_forward || qi_radialsteps != nr) {
278         if(qi_fft_forward) {
279             delete qi_fft_forward;
280             delete qi_fft_backward;
281         }
282         qi_radialsteps = nr;
283         qi_fft_forward = new kissfft<scalar_t>(nr*2, false);
284         qi_fft_backward = new kissfft<scalar_t>(nr*2, true);
285     }
286 }
```

10.10.3.2 void CPUTracker::ApplyOffsetGain (float * offset, float * gain, float offsetFactor, float gainFactor)

```

102 {
103     if (offset && !gain) {
104         for (int i=0;i<width*height;i++)
105             srcImage[i] = srcImage[i]+offset[i]*offsetFactor;
106     }
107     if (gain && !offset) {
108         for (int i=0;i<width*height;i++)
109             srcImage[i] = srcImage[i]*gain[i]*gainFactor;
110     }
111     if (gain && offset) {
112         for (int i=0;i<width*height;i++)
113             srcImage[i] = (srcImage[i]+offset[i]*offsetFactor)*gain[i]*gainFactor;
114     }
115 }
```

10.10.3.3 void CPUTracker::CalculateErrorCurve (double * errorcurve_dest, float * profile, float * zlut_sel)

```

806 {
807 #if 0
808     float* zlut_norm = ALLOCA_ARRAY(float, zlut_res);
809     float* prof_norm = ALLOCA_ARRAY(float, zlut_res);
810     for (int r=0;r<zlut_res;r++)
811         prof_norm[r] = rprof[r] * zlut_radialweights[r];
812     NormalizeRadialProfile(prof_norm, zlut_res);
813
814     for (int k=0;k<zlut_planes;k++) {
815         double diffsum = 0.0f;
816
817         if (zlut_radialweights.empty()) {
818             for (int r=0;r<zlut_res;r++) zlut_norm[r]=zlut_sel[k*zlut_res+r];
819         } else {
820             for (int r=0;r<zlut_res;r++) {
821                 zlut_norm[r] = zlut_sel[k*zlut_res+r] * zlut_radialweights[r];
822             }
823         }
824         NormalizeRadialProfile(zlut_norm, zlut_res);
825
826         for (int r = 0; r<zlut_res;r++) {
827             double d = prof_norm[r]-zlut_norm[r];
828             d = -d*d;
829             diffsum += d;
830         }
831         rprof_diff[k] = diffsum;
832     }
833 #else
834     for (int k=0;k<zlut_planes;k++) {
835         double diffsum = 0.0f;
836         for (int r = 0; r<zlut_res;r++) {
837             double d = profile[r]-zlut_sel[k*zlut_res+r];
838             if (!zlut_radialweights.empty())
839                 d*=zlut_radialweights[r];
840             d = -d*d;
841             diffsum += d;
842         }
843         errorcurve_dest[k] = diffsum;
844     }
845 #endif
846 }
```

10.10.3.4 float CPUTracker::CalculateErrorFlag (double * prof1, double * prof2)

```

863 {
864     //      sum(abs(expectedCurve-errorCurve))/size(errorCurve,2)
865     float flag = 0.0f;
866     for(int ii = 0; ii < zlut_res; ii++){
867         flag += std::abs(prof1[ii]-prof2[ii]);
868     }
869     return flag/zlut_res;
870 }
```

10.10.3.5 void CPUTracker::CalculateInterpolatedZLUTProfile (float * *profile_dest*, float *z*, int *zlutIndex*)

```

849 {
850     float* zlut_sel = GetRadialZLUT(zlutIndex);
851     for (int r = 0; r<zlut_res;r++) {
852         if ( z < 0 )
853             profile_dest[r] = 0;
854         else if( ((int)z+1) < zlut_planes ) // Index out of bounds if z > LUT resolution
855             profile_dest[r] = Lerp(zlut_sel[(int)z*zlut_res+r],zlut_sel[((int)z+1)*zlut_res+r],z-(int)z
856         );
857         else
858             profile_dest[r] = zlut_sel[(zlut_planes-1)*zlut_res+r];
859     }
860     NormalizeRadialProfile(profile_dest,zlut_res);

```

10.10.3.6 bool CPUTracker::CheckBoundaries (vector2f *center*, float *radius*)

```

186 {
187     return center.x + radius >= width ||
188         center.x - radius < 0 ||
189         center.y + radius >= height ||
190         center.y - radius < 0;
191 }

```

10.10.3.7 CPUTracker::Gauss2DResult CPUTracker::Compute2DGaussianMLE (vector2f *initial*, int *iterations*, float *sigma*)

```

526 {
527     vector2f pos = initial;
528     float I0 = mean*0.5f*width*height;
529     float bg = mean*0.5f;
530
531     const float _loSq2Sigma = 1.0f / (sqrtf(2) * sigma);
532     const float _loSq2PiSigma = 1.0f / (sqrtf(2*3.14159265359f) * sigma);
533     const float _loSq2PiSigma3 = 1.0f / (sqrtf(2*3.14159265359f) * sigma*sigma*sigma);
534
535     for (int i=0;i<iterations;i++)
536     {
537         double dL_dx = 0.0;
538         double dL_dy = 0.0;
539         double dL_dI0 = 0.0;
540         double dL_dIbg = 0.0;
541         double dL2_dx = 0.0;
542         double dL2_dy = 0.0;
543         double dL2_dI0 = 0.0;
544         double dL2_dIbg = 0.0;
545
546         double mu_sum = 0.0;
547
548         for (int y=0;y<height;y++)
549         {
550             for (int x=0;x<width;x++)
551             {
552                 float Xexp0 = (x-pos.x + .5f) * _loSq2Sigma;
553                 float Yexp0 = (y-pos.y + .5f) * _loSq2Sigma;
554
555                 float Xexp1 = (x-pos.x - .5f) * _loSq2Sigma;
556                 float Yexp1 = (y-pos.y - .5f) * _loSq2Sigma;
557
558                 float DeltaX = 0.5f * erf(Xexp0) - 0.5f * erf(Xexp1);
559                 float DeltaY = 0.5f * erf(Yexp0) - 0.5f * erf(Yexp1);
560                 float mu = bg + I0 * DeltaX * DeltaY;
561
562                 float dmu_dx = I0*_loSq2PiSigma * ( expf(-Xexp1*Xexp1) - expf(-Xexp0*Xexp0) ) * DeltaY;
563
564                 float dmu_dy = I0*_loSq2PiSigma * ( expf(-Yexp1*Yexp1) - expf(-Yexp0*Yexp0) ) * DeltaX;
565                 float dmu_dI0 = DeltaX*DeltaY;
566                 float dmu_dIbg = 1;
567
568                 float smp = GetPixel(x,y);
569                 float f = smp / mu - 1;
570                 dL_dx += dmu_dx * f;
571                 dL_dy += dmu_dy * f;
572                 dL_dI0 += dmu_dI0 * f;

```

```

573         dL_dIbg += dmu_dIbg * f;
574
575         float d2mu_dx = I0*_loSq2PiSigma3 * ( (x - pos.x - .5f) * expf (-Xexp1*Xexp1) - (x - pos.
576 x + .5f) * expf (-Xexp0*Xexp0) ) * DeltaY;
577         float d2mu_dy = I0*_loSq2PiSigma3 * ( (y - pos.y - .5f) * expf (-Yexp1*Yexp1) - (y - pos.
578 y + .5f) * expf (-Yexp0*Yexp0) ) * DeltaX;
579         dL2_dx += d2mu_dx * f - dmu_dx*dmu_dx * smp / (mu*mu);
580         dL2_dy += d2mu_dy * f - dmu_dy*dmu_dy * smp / (mu*mu);
581         dL2_dI0 += -dmu_dI0*dmu_dI0 * smp / (mu*mu);
582         dL2_dIbg += -smp / (mu*mu);
583
584     }
585
586     double mean_mu = mu_sum / (width*height);
587
588     pos.x -= dL_dx / dL2_dx;
589     pos.y -= dL_dy / dL2_dy;
590     I0 -= dL_dI0 / dL2_dI0;
591     bg -= dL_dIbg / dL2_dIbg;
592 }
593
594 Gauss2DResult r;
595 r.pos = pos;
596 r.I0 = I0;
597 r.bg = bg;
598
599 return r;
600 }
```

10.10.3.8 float CPUTracker::ComputeAsymmetry (vector2f center, int radialSteps, int angularSteps, float minRadius, float maxRadius, float * dstAngProf = 0)

```

605 {
606     vector2f* radialDirs = (vector2f*)ALLOCA(sizeof(
607     vector2f)*angularSteps);
608     for (int j=0;j<angularSteps;j++) {
609         float ang = 2*3.141593f*j/(float)angularSteps;
610         radialDirs[j] = vector2f (cosf(ang), sinf(ang));
611     }
612
613     // profile along single radial direction
614     float* rline = (float*)ALLOCA(sizeof(float)*radialSteps);
615
616     if (!dstAngProf)
617         dstAngProf = (float*)ALLOCA(sizeof(float)*radialSteps);
618
619     float rstep = (maxRadius-minRadius) / radialSteps;
620
621     for (int a=0;a<angularSteps;a++) {
622         // fill rline
623         // angularProfile[a] = rline COM
624
625         float r = minRadius;
626         for (int i=0;i<radialSteps;i++) {
627             float x = center.x + radialDirs[a].x * r;
628             float y = center.y + radialDirs[a].y * r;
629             rline[i] = Interpolate(srcImage,width,height, x,y);
630             r += rstep;
631         }
632
633         // Compute rline COM
634         dstAngProf[a] = ComputeBgCorrectedCOM1D(rline, radialSteps, 1.0f);
635     }
636
637     float stdev = ComputeStdDev(dstAngProf, angularSteps);
638 }
```

10.10.3.9 vector2f CPUTracker::ComputeMeanAndCOM (float bgcorrection = 0.0f)

```

690 {
691     double sum=0, sum2=0;
692     double momentX=0;
693     double momentY=0;
```

```

694     // Order is important
695     //COM: x:53.959133, y:53.958984
696     //COM: x:53.958984, y:53.959133
697
698     for (int y=0;y<height;y++) {
699         for (int x=0;x<width;x++) {
700             float v = GetPixel(x,y);
701             sum += v;
702             sum2 += v*v;
703         }
704     }
705
706     float invN = 1.0f/(width*height);
707     mean = sum * invN;
708     stdev = sqrtf(sum2 * invN - mean * mean);
709     sum = 0.0f;
710
711     /*float *ymom = ALLOCA_ARRAY(float, width);
712     float *xmom = ALLOCA_ARRAY(float, height);
713
714     for(int x=0;x<width;x++)
715         ymom[x]=0;
716     for(int y=0;y<height;y++)
717         xmom[y]=0; */
718
719     // Comments: Gaussian mask not currently used
720     //vector2f centre = vector2f((float)width/2,(float)height/2);
721     //float sigma = (float)width/4; // Suppress to 5% at edges
722     //float devi = 1/(2*sigma*sigma);
723
724     for(int x=0;x<width;x++) {
725         for (int y=0;y<height;y++) {
726             float v = GetPixel(x,y);
727
728             //float xabs = (x-centre.x); float yabs = (y-centre.y);
729             //float gaussfact = 1.0f;//expf(- xabs*xabs*devi - yabs*yabs*devi);
730
731             v = std::max(0.0f, fabs(v-mean)-bgcorrection*stdev);
732             sum += v;
733             // xmom[y] += x*v;
734             // ymom[x] += y*v;
735             momentX += x*(double)v;
736             momentY += y*(double)v;
737         }
738     }
739
740     vector2f com;
741     com.x = (float)( momentX / sum );
742     com.y = (float)( momentY / sum );
743     return com;
744 }
```

10.10.3.10 `vector2f CPUTracker::ComputeQI (vector2f initial, int iterations, int radialSteps, int angularStepsPerQuadrant, float angStepIterationFactor, float minRadius, float maxRadius, bool & boundaryHit, float * radialweights = 0)`

```

290 {
291     int nr=radialSteps;
292 #ifdef _DEBUG
293     std::copy(srcImage, srcImage+width*height,
294               debugImage);
295     maxValue = *std::max_element(srcImage,srcImage+width*
296                                 height);
297 #endif
298     if (angularStepsPerQ != quadrantDirs.size()) {
299         quadrantDirs.resize(angularStepsPerQ);
300         for (int j=0;j<angularStepsPerQ;j++) {
301             float ang = 0.5*3.141593f*(j+0.5f)/(float)angularStepsPerQ;
302             quadrantDirs[j] = vector2f( cosf(ang), sinf(ang) );
303         }
304     }
305     AllocateQIFFTs(radialSteps);
306
307     scalar_t* buf = (scalar_t*)ALLOCA(sizeof(scalar_t)*nr*4);
308     scalar_t* q0=buf, *q1=buf+nr, *q2=buf+nr*2, *q3=buf+nr*3;
309     complex_t* concat0 = (complex_t*)ALLOCA(sizeof(
310                                         complex_t)*nr*2);
311     complex_t* concat1 = concat0 + nr;
312     vector2f center = initial;
```

```

313     float pixelsPerProfLen = (maxRadius-minRadius)/radialSteps;
314     boundaryHit = false;
315
316     float angsteps = angularStepsPerQ / powf(angStepIterationFactor, iterations);
317
318     for (int k=0;k<iterations;k++) {
319         // check bounds
320         boundaryHit = CheckBoundaries(center, maxRadius);
321
322         for (int q=0;q<4;q++) {
323             ComputeQuadrantProfile(buf+q*nr, nr, angsteps, q, minRadius, maxRadius,
324             center, radialWeights);
325             //NormalizeRadialProfile(buf+q*nr, nr);
326         }
327 #ifdef QI_DEBUG
328         cmp_cpu_qi_prof.assign (buf,buf+4*nr);
329 #endif
330
331         // Build Ix = [ qL(-r)  qR(r) ]
332         // qL = q1 + q2   (concat0)
333         // qR = q0 + q3   (concat1)
334         for(int r=0;r<nr;r++) {
335             concat0[nr-r-1] = q1[r]+q2[r];
336             concat1[r] = q0[r]+q3[r];
337         }
338
339         scalar_t offsetX = QI_ComputeOffset(concat0, nr, 0);
340
341         // Build Iy = [ qB(-r)  qT(r) ]
342         // qT = q0 + q1
343         // qB = q2 + q3
344         for(int r=0;r<nr;r++) {
345             concat1[r] = q0[r]+q1[r];
346             concat0[nr-r-1] = q2[r]+q3[r];
347         }
348
349         scalar_t offsetY = QI_ComputeOffset(concat0, nr, 1);
350
351 #ifdef QI_DBG_EXPORT
352         std::copy(concat0, concat0+nr*2,tmp.begin()+nr*2);
353         WriteComplexImageAsCSV("cpuprofxy.txt", &tmp[0], nr, 4);
354         dbprintf("[%d] OffsetX: %f, OffsetY: %f\n", k, offsetX, offsetY);
355 #endif
356
357         center.x += offsetX * pixelsPerProfLen;
358         center.y += offsetY * pixelsPerProfLen;
359
360         angsteps *= angStepIterationFactor;
361     }
362
363     return center;
364 }
```

10.10.3.11 void CPUTracker::ComputeQuadrantProfile (scalar_t * dst, int radialSteps, int angularSteps, int quadrant, float minRadius, float maxRadius, vector2f center, float * radialWeights = 0)

```

643 {
644     const int qmat[] = {
645         1, 1,
646         -1, 1,
647         -1, -1,
648         1, -1 };
649     int mx = qmat[2*quadrant+0];
650     int my = qmat[2*quadrant+1];
651
652     if (angularSteps < MIN_RADPROFILE_SMP_COUNT)
653         angularSteps = MIN_RADPROFILE_SMP_COUNT;
654
655     for (int i=0;i<radialSteps;i++)
656         dst[i]=0.0f;
657
658     scalar_t total = 0.0f;
659     scalar_t rstep = (maxRadius - minRadius) / radialSteps;
660     for (int i=0;i<radialSteps; i++) {
661         scalar_t sum = 0.0f;
662         scalar_t r = minRadius + rstep * i;
663
664         int nPixels = 0;
665         scalar_t angstepf = (scalar_t) quadrantDirs.size() / angularSteps;
666         for (int a=0;a<angularSteps;a++) {
```

```

667         int i = (int)angstepf * a;
668         scalar_t x = center.x + mx*quadrantDirs[i].x * r;
669         scalar_t y = center.y + my*quadrantDirs[i].y * r;
670
671         bool outside;
672         scalar_t v = Interpolate(srcImage, width,
673             height, x, y, &outside);
674         if (!outside) {
675             sum += v;
676             nPixels++;
677             MARKPIXELI(x, y);
678         }
679     }
680     dst[i] = nPixels>=MIN_RADPROFILE_SMP_COUNT ? sum/nPixels :
681     mean;
682     if (radialWeights) dst[i] *= radialWeights[i];
683     total += dst[i];
684 }
685 //WriteImageAsCSV(SPrintf("D:\\TestImages\\qprof%d.txt", quadrant).c_str(), dst, 1, radialSteps);
686 //WriteArrayAsCSVRow("D:\\TestImages\\radialWeights.csv", radialWeights, radialSteps, false);
687 }

```

10.10.3.12 void CPUTracker::ComputeRadialProfile (float * dst, int radialSteps, int angularSteps, float minradius, float maxradius, vector2f center, bool crp, bool * boundaryHit = 0, bool normalize = true)

```

756 {
757     bool boundaryHit = CheckBoundaries(center, maxradius);
758     if (pBoundaryHit) *pBoundaryHit = boundaryHit;
759
760     ImageData imgData (srcImage, width, height);
761     if (crp) {
762         ComputeCRP(dst, radialSteps, angularSteps, minradius, maxradius, center, &imgData,
763         mean);
764     } else {
765         ::ComputeRadialProfile(dst, radialSteps, angularSteps, minradius, maxradius,
766         center, &imgData, mean, normalize);
767     }

```

10.10.3.13 vector2f CPUTracker::ComputeXCorInterpolated (vector2f initial, int iterations, int profileWidth, bool & boundaryHit)

```

194 {
195     // extract the image
196     vector2f pos = initial;
197
198     if (!xcorBuffer)
199         xcorBuffer = new XCor1DBuffer(xcorw);
200
201     if (xcorw < profileWidth)
202         profileWidth = xcorw;
203
204 #ifdef _DEBUG
205     std::copy(srcImage, srcImage+width*height,
206               debugImage);
207     maxImageValue = *std::max_element(srcImage, srcImage+width*
208                                     height);
209 #endif
210
211     if (xcorw > width || xcorw > height) {
212         boundaryHit = true;
213         return initial;
214     }
215
216     complex_t* prof = (complex_t*)ALLOCA(sizeof(
217     complex_t)*xcorw);
218     complex_t* prof_rev = (complex_t*)ALLOCA(sizeof(
219     complex_t)*xcorw);
220     scalar_t* prof_autocor = (scalar_t*)ALLOCA(sizeof(
221     scalar_t)*xcorw);
222
223     boundaryHit = false;
224     for (int k=0;k<iterations;k++) {

```

```

220     boundaryHit = CheckBoundaries(pos, XCorScale*
221         xcorw/2);
222     float xmin = pos.x - XCorScale * xcorw/2;
223     float ymin = pos.y - XCorScale * xcorw/2;
224
225     // generate X position xcor array (summing over y range)
226     for (int x=0;x<xcorw;x++) {
227         scalar_t s = 0.0f;
228         int n=0;
229         for (int y=0;y<profileWidth;y++) {
230             scalar_t xp = x * XCorScale + xmin;
231             scalar_t yp = pos.y + XCorScale * (y - profileWidth/2);
232             bool outside;
233             s += Interpolate(srcImage, width, height, xp, yp, &outside);
234             n += outside?0:1;
235             MARKPIXELI(xp, yp);
236         }
237         if (n >0) s/=n;
238         else s=0;
239         prof [x] = s;
240         prof_rev [xcorw-x-1] = s;
241     }
242
243     xcorBuffer->XCorFFTHelper(prof, prof_rev, prof_autocor);
244     scalar_t offsetX =
245     ComputeMaxInterp<scalar_t,QI_LSQFIT_NWEIGHTS>::Compute
246     (prof_autocor, xcorw, QIWeights) - (scalar_t)xcorw/2;
247
248     // generate Y position xcor array (summing over x range)
249     for (int y=0;y<xcorw;y++) {
250         scalar_t s = 0.0f;
251         int n=0;
252         for (int x=0;x<profileWidth;x++) {
253             scalar_t xp = pos.x + XCorScale * (x - profileWidth/2);
254             scalar_t yp = y * XCorScale + ymin;
255             bool outside;
256             s += Interpolate(srcImage,width,height, xp, yp, &outside);
257             n += outside?0:1;
258             MARKPIXELI(xp,yp);
259         }
260         if (n >0) s/=n;
261         else s=0;
262         prof[y] = s;
263         prof_rev [xcorw-y-1] = s;
264     }
265
266     xcorBuffer->XCorFFTHelper(prof,prof_rev, prof_autocor);
267     //WriteImageAsCSV("xcorautoconv.txt",&xcorBuffer->Y_result[0],xcorBuffer->Y_result.size(),1);
268     scalar_t offsetY =
269     ComputeMaxInterp<scalar_t, QI_LSQFIT_NWEIGHTS>::Compute
270     (prof_autocor, xcorw, QIWeights) - (scalar_t)xcorw/2;
271
272     pos.x += (offsetX - 1) * XCorScale * 0.5f;
273     pos.y += (offsetY - 1) * XCorScale * 0.5f;
274 }
275
276 return pos;
277 }
```

10.10.3.14 float CPUTracker::ComputeZ(vector2f center, int angularSteps, int zlutIndex, bool * boundaryHit = 0, float * profile = 0, float * cmpprof = 0, bool normalizeProfile = true) [inline]

```

98     {
99         float* prof = profile ? profile : ALLOCA_ARRAY(float,
100             zlut_res);
101         ComputeRadialProfile(prof,zlut_res,angularSteps,
102             zlut_minradius, zlut_maxradius, center, false, boundaryHit, normalizeProfile);
103         return LUTProfileCompare(prof, zlutIndex, cmpprof,
104             LUTProfMaxQuadraticFit);
105     }
```

10.10.3.15 void CPUTracker::FourierRadialProfile(float * dst, int radialSteps, int angularSteps, float minradius, float maxradius)

```
1090 {
```

```

1091     FourierTransform2D();
1092     ImageData img(srcImage, width, height);
1093     ::ComputeRadialProfile(dst, radialSteps, angularSteps, 1,
1094         width*0.3f, vector2f(width/2, height/2), &img, 0, false);
1095     for (int i=0;i<radialSteps;i++) {
1096         dst[i]=logf(dst[i]);
1097     }
1098 } //NormalizeRadialProfile(dst, radialSteps);

```

10.10.3.16 void CPUTracker::FourierTransform2D()

```

1046 {
1047     //kissfft<float> yfft(h, inverse);
1048     if (!fft2d){
1049         fft2d = new FFT2D(width, height);
1050     }
1051     fft2d->Apply(srcImage);
1052 }

```

10.10.3.17 float* CPUTracker::GetDebugImage() [inline]

```
119 { return debugImage; }
```

10.10.3.18 int CPUTracker::GetHeight() [inline]

```
66 { return height; }
```

10.10.3.19 float& CPUTracker::GetPixel(int x, int y) [inline]

```
64 { return srcImage[width*y+x]; }
```

10.10.3.20 float* CPUTracker::GetRadialZLUT(int index) [inline]

```
47 { return &zluts[zlut_res*zlut_planes*index]; }
```

10.10.3.21 int CPUTracker::GetWidth() [inline]

```
65 { return width; }
```

10.10.3.22 bool CPUTracker::KeepInsideBoundaries (*vector2f* * *center*, float *radius*)

```

165 {
166     bool boundaryHit = center->x + radius >= width ||
167         center->x - radius < 0 ||
168         center->y + radius >= height ||
169         center->y - radius < 0;
170
171     if (center->x - radius < 0.0f)
172         center->x = radius;
173
174     if (center->y - radius < 0.0f)
175         center->y = radius;
176
177     if (center->x + radius >= width)
178         center->x = width-radius-1;
179
180     if (center->y + radius >= height)
181         center->y = height-radius-1;
182
183     return boundaryHit;
184 }
```

10.10.3.23 float CPUTracker::LUTProfileCompare (float * *profile*, int *zlutIndex*, float * *cmpProf*,
LUTProfileMaxComputeMode *maxPosMethod*, float * *lsqfittedcurve* = 0, int * *maxPos* = 0, int *frameNum* = 0
)

```

873 {
874     /* From this function save:
875      - Frame number?
876
877      - Original image
878      - Profile
879      - LUT
880      - Error curve
881      - Final value
882 */
883
884     bool freeMem = false;
885     std::string outputName;
886     float zOffset;
887
888     if(testRun){
889         outputName = Sprintf("%s%05d-%05d", GetCurrentOutputPath().c_str(),
890                             zlutIndex, frameNum);
891
892         if(FILE *file = fopen(Sprintf("%s.jpg",outputName.c_str()).c_str(), "r")){
893             fclose(file);
894             printf("File exists\n");
895         }
896         SaveImage(Sprintf("%s.jpg",outputName.c_str()).c_str());
897         //FloatToJPEGFile(Sprintf("%s-prof.jpg",outputName.c_str()).c_str(), rprof, zlut_res, 1);
898         WriteArrayAsCSVRow(Sprintf("%s-prof.txt",outputName.c_str()).c_str(),
899                             rprof, zlut_res, 0);
899
900         if(!fitcurve){
901             fitcurve = new float[zlut_planes];
902             freeMem = true;
903         }
904
905         /* For testing, we want to introduce a fixed height offset of xx nm, regardless of amount of zlut
906         planes
907             Z is in zlut planes here, so calculate back
908
909             int numImgInStack = 1218;
910             int numPositions = 1001; // ~10nm/frame
911             float range = 10.0f; // total range 25.0 um -> 35.0 um
912             float umPerImg = range/numImgInStack; //
913
914             int zplanes = 50;
915             int startFrame = 400;
916
917             float range = 10.0f; // total range 25.0 um -> 35.0 um
918             int numImgInStack = 1218;
919             float umPerImg = range/numImgInStack;
920             int startFrame = 400;
921             int usedFrames = numImgInStack-startFrame;
922             float rangeInLUT = usedFrames*umPerImg; // Total um of LUT range
923             float umPerPlane = rangeInLUT/zlut_planes;
```

```

924         float error = 0.000f; // Forced error in um
925         zOffset = error/umPerPlane; // um * planes / um
926     }
927
928     if (!zluts)
929         return 0.0f;
930
931     double* rprof_diff = ALLOCA_ARRAY(double, zlut_planes);
932     //WriteImageAsCSV("zlutradprof-cpu.txt", rprof, zlut_res, 1);
933
934     // Now compare the radial profile to the profiles stored in Z
935     float* zlut_sel = GetRadialZLUT(zlutIndex);
936
937     if(testRun){
938         FloatToJPEGFile(SPrintf("%s-zlut.jpg",outputName.c_str()).c_str(), zlut_sel,
939                         zlut_res, zlut_planes);
940     }
941
942     CalculateErrorCurve(rprof_diff, rprof, zlut_sel);
943
944     if (cmpProf) {
945         //cmpProf->resize(zlut_planes);
946         std::copy(rprof_diff, rprof_diff+zlut_planes, cmpProf);
947     }
948
949     if (maxPosComputeMode == LUTProfMaxQuadraticFit) {
950         if (maxPos) {
951             *maxPos = std::max_element(rprof_diff, rprof_diff+zlut_planes) - rprof_diff;
952         }
953
954         if (fitcurve) {
955             LsqSqQuadFit<double> fit;
956             float z = ComputeMaxInterp<double, ZLUT_LSQFIT_NWEIGHTS>::Compute
957             (rprof_diff, zlut_planes, ZLUTWeights_d, &fit);
958
959             int iMax = std::max_element(rprof_diff, rprof_diff+zlut_planes) - rprof_diff;
960             for (int i=0;i<zlut_planes;i++)
961                 fitcurve[i] = fit.compute(i-iMax);
962
963             if(testRun) {
964                 z += zOffset; // For testing purposes
965
966                 // Calculate errorcurve of found ZLUT plane
967                 float* int_prof = ALLOCA_ARRAY(float, zlut_res);
968                 CalculateInterpolatedZLUTProfile(int_prof, z, zlutIndex);
969
970                 double* plane_auto_diff = ALLOCA_ARRAY(double, zlut_planes);
971                 CalculateErrorCurve(plane_auto_diff, int_prof, zlut_sel);
972                 /*for (int k=0;k<zlut_planes;k++) {
973                     double diffsum = 0.0f;
974                     for (int r = 0; r<zlut_res;r++) {
975                         double d = int_prof[r] - zlut_sel[k*zlut_res+r];
976                         if (!zlut_radialweights.empty())
977                             d*=zlut_radialweights[r];
978                         d = -d*d;
979                         diffsum += d;
980                     }
981                     plane_auto_diff[k] = diffsum;
982                 }*/
983
984                 //float errorFlag = CalculateErrorFlag(rprof_diff, plane_auto_diff);
985
986                 WriteArrayAsCSVRow(SPrintf("%s-int-prof.txt",outputName.c_str()).
987                                     c_str(), int_prof, zlut_res, 0);
988
989                 FILE* f = fopen(SPrintf("%s-out.txt",outputName.c_str()).c_str(),"w+");
990                 fprintf_s(f,"z: %f\n",z);
991                 for (int i=0;i<zlut_planes;i++)
992                     fprintf_s(f,"%f\t%f\t%f\t%f\t%f\n",rprof_diff[i],fitcurve[i],plane_auto_diff[i],fit.
993 a,fit.vertexForm());
994                 fclose(f);
995
996                 if(freeMem)
997                     delete[] fitcurve;
998
999             }
1000
1001             return z;
1002         } else {
1003             return ComputeMaxInterp<double, ZLUT_LSQFIT_NWEIGHTS>::Compute
1004             (rprof_diff, zlut_planes, ZLUTWeights_d);
1005         }
1006         else if (maxPosComputeMode == LUTProfMaxSimpleInterp) {
1007             assert(0);
1008             return 0;
1009         } else
1010             return ComputeSplineFitMaxPos(rprof_diff, zlut_planes);

```

```
1006 }
```

10.10.3.24 float CPUTracker::LUTProfileCompareAdjustedWeights (float * rprof, int zlutIndex, float z_estim)

```
1009 {
1010     if (!zluts)
1011         return 0.0f;
1012
1013     double* rprof_diff = ALLOCA_ARRAY(double, zlut_planes);
1014
1015     // Now compare the radial profile to the profiles stored in Z
1016     float* zlut_sel = GetRadialZLUT(zlutIndex);
1017
1018     int zi = (int)z_estim;
1019     if (zi > zlut_planes-2) zi = zlut_planes-2;
1020     float* z0 = &zlut_sel[zi*zlut_res];
1021     float* z1 = &zlut_sel[(zi+1)*zlut_res];
1022     double* zlut_weights = ALLOCA_ARRAY(double, zlut_res);
1023     for (int i=0;i<zlut_res;i++)
1024         zlut_weights[i] = fabs((double)z1[i]-(double)z0[i]) * ( zlut_minradius +
1025             zlut_maxradius - zlut_minradius ) * i/(float)zlut_res );
1026     for (int k=0;k<zlut_planes;k++) {
1027         double diffsum = 0.0f;
1028         for (int r = 0; r<zlut_res;r++) {
1029             double d = (double)rprof[r]-(double)zlut_sel[k*zlut_res+r];
1030             d = -d*d;
1031             d *= zlut_weights[r];
1032             diffsum += d;
1033         }
1034         rprof_diff[k] = diffsum;
1035     }
1036     //return ComputeSplineFitMaxPos(rprof_diff,zlut_planes);
1037     return ComputeMaxInterp<double, ZLUT_LSQFIT_NWEIGHTS>::Compute
1038     (rprof_diff, zlut_planes, ZLUTWeights_d);
1038 }
```

10.10.3.25 void CPUTracker::Normalize (float * image = 0)

```
748 {
749     if (!d) d=srcImage;
750     normalize(d, width, height);
751 }
```

10.10.3.26 scalar_t CPUTracker::QI_ComputeOffset (complex_t * qi_profile, int nr, int axisForDebug)

```
380 {
381     complex_t* reverse = ALLOCA_ARRAY(complex_t, nr*2);
382     complex_t* fft_out = ALLOCA_ARRAY(complex_t, nr*2);
383     complex_t* fft_out2 = ALLOCA_ARRAY(complex_t, nr*2);
384
385     for(int x=0;x<nr*2;x++)
386         reverse[x] = profile[nr*2-1-x];
387
388     qi_fft_forward->transform(profile, fft_out);
389     qi_fft_forward->transform(reverse, fft_out2); // fft_out2 contains
390     fourier-domain version of reverse profile
391
392     // multiply with conjugate
393     for(int x=0;x<nr*2;x++)
394         fft_out[x] *= conjugate(fft_out2[x]);
395
396     qi_fft_backward->transform(fft_out, fft_out2);
397
398 #ifdef QI_DEBUG
399     cmp_cpu_qi_fft_out.assign(fft_out2, fft_out2+nr*2);
400 #endif
401
402     // fft_out2 now contains the autoconvolution
403     // convert it to float
404     scalar_t* autoconv = ALLOCA_ARRAY(scalar_t, nr*2);
```

```

404     for(int x=0;x<nr*2;x++)  {
405         autoconv[x] = fft_out2[ (x+nr)% (nr*2) ].real();
406     }
407
408     float* profileReal = ALLOCA_ARRAY(float, nr*2);
409     for(int x=0;x<nr*2;x++)  {
410         profileReal[x] = profile[x].real();
411     }
412     //WriteArrayAsCSVRow(SPrintf("D:\\TestImages\\AutoconvProf-%d.txt",axisForDebug).c_str()
413     ,profileReal,nr*2,false);
413     //WriteArrayAsCSVRow(SPrintf("D:\\TestImages\\Autoconv-%d.txt",axisForDebug).c_str()
414     ,autoconv,nr*2,false); // */
415
415     scalar_t maxPos =
416     ComputeMaxInterp<scalar_t, QI_LSQFIT_NWEIGHTS>::Compute
417     (autoconv, nr*2, QIWeights);
416     return (maxPos - nr) * (3.14159265359f / 4);
417 }
```

10.10.3.27 vector3f CPUTracker::QuadrantAlign (vector3f *initial*, int *beadIndex*, int *angularStepsPerQuadrant*, bool & *boundaryHit*)

```

464 {
465     float* zlut = GetRadialZLUT(beadIndex);
466     int res=zlut_res;
467     complex_t* profile = ALLOCA_ARRAY(complex_t, res*2);
468     complex_t* concat0 = ALLOCA_ARRAY(complex_t, res*2);
469     complex_t* concat1 = concat0 + res;
470
471     memset(concat0, 0, sizeof(complex_t)*res*2);
472
473     int zp0 = clamp( (int)pos.z, 0, zlut_planes - 1);
474     int zp1 = clamp( (int)pos.z + 1, 0, zlut_planes - 1);
475
476     float* zlut0 = &zlut[ res * zp0 ];
477     float* zlut1 = &zlut[ res * zp1 ];
478     float frac = pos.z - (int)pos.z;
479     for (int r=0;r<zlut_res;r++) {
480         // Interpolate plane
481         double zlutValue = Lerp(zlut0[r], zlut1[r], frac);
482         concat0[res-r-1] = concat1[r] = zlutValue;
483     }
484 // WriteComplexImageAsCSV("qa_zlutprof.txt", concat0, res,2);
485 qa_fft_forward->transform(concat0, profile);
486
487     scalar_t* buf = ALLOCA_ARRAY(scalar_t, res*4);
488     scalar_t* q0=buf, *q1=buf+res, *q2=buf+res*2, *q3=buf+res*3;
489
490     boundaryHit = CheckBoundaries(vector2f(pos.x,pos.y),
491     zlut_maxradius);
491     for (int q=0;q<4;q++) {
492         scalar_t *quadrantProfile = buf+q*res;
493         ComputeQuadrantProfile(quadrantProfile, res, angularStepsPerQuadrant, q,
494     zlut_minradius, zlut_maxradius, vector2f(pos.x,pos.y));
495         NormalizeRadialProfile(quadrantProfile, res);
496     }
497
498     //WriteImageAsCSV("qa_qdr.txt" , buf, res,4);
499     float pixelsPerProfLen = (zlut_maxradius-zlut_minradius)/zlut_res;
500     boundaryHit = false;
501
502     // Build Ix = [ qL(-r)  qR(r) ]
503     // qL = q1 + q2   (concat0)
504     // qR = q0 + q3   (concat1)
505     for(int r=0;r<res;r++) {
506         concat0[res-r-1] = q1[r]+q2[r];
507         concat1[r] = q0[r]+q3[r];
508     }
509
510     scalar_t offsetX = QuadrantAlign_ComputeOffset(concat0, profile, res
511     , 0);
512
512     // Build Iy = [ qB(-r)  qT(r) ]
513     // qT = q0 + q1
514     // qB = q2 + q3
515     for(int r=0;r<res;r++) {
516         concat1[r] = q0[r]+q1[r];
517         concat0[res-r-1] = q2[r]+q3[r];
518     }
519 }
```

```

520     scalar_t offsetY = QuadrantAlign_ComputeOffset(concat0, profile, res
521     , 1);
522     return vector3f(pos.x + offsetX * pixelsPerProfLen, pos.y + offsetY * pixelsPerProfLen, pos.z);
523 }

```

10.10.3.28 scalar_t CPUTracker::QuadrantAlign_ComputeOffset (complex_t * *profile*, complex_t * *zlut_prof_fft*, int *nr*, int *axisForDebug*)

```

422 {
423     complex_t* fft_out = ALLOCA_ARRAY(complex_t, nr*2);
424
425 // WriteComplexImageAsCSV("qa_profile.txt", profile, nr*2, 1);
426
427 qa_fft_forward->transform(profile, fft_out);
428
429 // multiply with conjugate
430 for(int x=0;x<nr*2;x++)
431     fft_out[x] *= conjugate(zlut_prof_fft[x]);
432
433 qa_fft_backward->transform(fft_out, profile);
434
435 #ifdef QI_DEBUG
436     cmp_cpu_qi_fft_out.assign(fft_out2, fft_out2+nr*2);
437 #endif
438
439 // profile now contains the autoconvolution
440 // convert it to float
441 scalar_t* autoconv = ALLOCA_ARRAY(scalar_t, nr*2);
442 for(int x=0;x<nr*2;x++) {
443     autoconv[x] = profile[(x+nr)%nr*2].real();
444 }
445
446 // WriteImageAsCSV("qa_autoconv.txt", autoconv, nr*2, 1);
447
448 scalar_t maxPos =
449 ComputeMaxInterp<scalar_t, QI_LSQFIT_NWEIGHTS>::Compute
450 (autoconv, nr*2, QIWeights);
451 return (maxPos - nr) * (3.14159265359f / 4);
452 }

```

10.10.3.29 void CPUTracker::SaveImage (const char * *filename*)

```

1041 {
1042     FloatToJPEGFile(filename, srcImage, width,
1043     height);
1044 }

```

10.10.3.30 template<typename TPixel > void CPUTracker::SetImage (TPixel * *srcImage*, uint *srcpitch*)

```

132 {
133     uchar* bp = (uchar*)data;
134
135     for (int y=0;y<height;y++) {
136         for (int x=0;x<width;x++) {
137             srcImage[y*width+x] = ((TPixel*)bp)[x];
138         }
139         bp += pitchInBytes;
140     }
141
142     mean=0.0f;
143 }

```

10.10.3.31 void CPUTracker::SetImage16Bit (ushort * *srcImage*, uint *srcpitch*) [inline]

```

88 { SetImage(srcImage, srcpitch); }

```

10.10.3.32 void CPUTracker::SetImage8Bit(uchar * *srcImage*, uint *srcpitch*) [inline]

```
89 { SetImage(srcImage, srcpitch); }
```

10.10.3.33 void CPUTracker::SetImageFloat(float * *srcImage*)

```
95 {
96     for (int k=0;k<width*height;k++)
97         srcImage[k]=src[k];
98     mean=0.0f;
99 }
```

10.10.3.34 void CPUTracker::SetRadialWeights(float * *w*)

```
797 {
798     if (radweights) {
799         zlut_radialweights.resize(zlut_res);
800         std::copy(radweights, radweights+zlut_res, zlut_radialweights.begin());
801     } else
802         zlut_radialweights.clear();
803 }
```

10.10.3.35 void CPUTracker::SetRadialZLUT(float * *data*, int *planes*, int *res*, int *num_zluts*, float *minradius*, float *maxradius*, bool *copyMemory*, bool *useCorrelation*)

```
770 {
771     if (zluts && zlut_memoryOwner)
772         delete[] zluts;
773
774     if (copyMemory) {
775         zluts = new float[planes*res*numLUTs];
776         std::copy(data, data+(planes*res*numLUTs), zluts);
777     } else
778         zluts = data;
779     zlut_memoryOwner = copyMemory;
780     zlut_planes = planes;
781     zlut_res = res;
782     zlut_count = numLUTs;
783     zlut_minradius = minradius;
784     zlut_maxradius = maxradius;
785     zlut_useCorrelation = useCorrelation;
786
787     if (qa_fft_backward) {
788         delete qa_fft_backward;
789         delete qa_fft_forward;
790     }
791
792     qa_fft_forward = new kissfft<scalar_t>(res*2, false);
793     qa_fft_backward = new kissfft<scalar_t>(res*2, true);
794 }
```

10.10.4 Member Data Documentation

10.10.4.1 float * CPUTracker::debugImage

10.10.4.2 FFT2D* CPUTracker::fft2d

10.10.4.3 int CPUTracker::height

- 10.10.4.4 float CPUTracker::mean
- 10.10.4.5 kissfft<scalar_t> * CPUTracker::qa_fft_backward
- 10.10.4.6 kissfft<scalar_t> * CPUTracker::qa_fft_forward
- 10.10.4.7 kissfft<scalar_t> * CPUTracker::qi_fft_backward
- 10.10.4.8 kissfft<scalar_t> * CPUTracker::qi_fft_forward
- 10.10.4.9 int CPUTracker::qi_radialsteps
- 10.10.4.10 std::vector<vector2f> CPUTracker::quadrantDirs
- 10.10.4.11 std::vector<vector2f> CPUTracker::radialDirs
- 10.10.4.12 float* CPUTracker::srclImage
- 10.10.4.13 float CPUTracker::stdev
- 10.10.4.14 bool CPUTracker::testRun
- 10.10.4.15 int CPUTracker::trackerID
- 10.10.4.16 int CPUTracker::width
- 10.10.4.17 XCor1DBuffer* CPUTracker::xcorBuffer
- 10.10.4.18 int CPUTracker::xcorw
- 10.10.4.19 int CPUTracker::zlut_count
- 10.10.4.20 float CPUTracker::zlut_maxradius
- 10.10.4.21 bool CPUTracker::zlut_memoryOwner
- 10.10.4.22 float CPUTracker::zlut_minradius
- 10.10.4.23 int CPUTracker::zlut_planes
- 10.10.4.24 std::vector<float> CPUTracker::zlut_radialweights
- 10.10.4.25 int CPUTracker::zlut_res
- 10.10.4.26 bool CPUTracker::zlut_useCorrelation
- 10.10.4.27 float* CPUTracker::zluts

The documentation for this class was generated from the following files:

- cputrack/cpu_tracker.h
- cputrack/cpu_tracker.cpp

10.11 CUDADeviceInfo Struct Reference

```
#include <lv_qtrk_api.h>
```

Public Attributes

- LStrHandle `name`
- int `clockRate`
- int `multiProcCount`
- int `major`
- int `minor`

10.11.1 Member Data Documentation

10.11.1.1 int CUDADeviceInfo::clockRate

10.11.1.2 int CUDADeviceInfo::major

10.11.1.3 int CUDADeviceInfo::minor

10.11.1.4 int CUDADeviceInfo::multiProcCount

10.11.1.5 LStrHandle CUDADeviceInfo::name

The documentation for this struct was generated from the following file:

- cputrack/[lv_qtrk_api.h](#)

10.12 cudalmageList< T > Struct Template Reference

```
#include <cudaImageList.h>
```

Public Types

- enum { `MaxImageWidth` = 8192 }

Public Member Functions

- `CUBOTH int fullwidth ()`
- `CUBOTH int fullheight ()`
- `CUBOTH int capacity ()`
- `CUBOTH int numpixels ()`
- `CUBOTH bool isEmpty ()`
- template<int Flags>
 `void allocateHostImageBuffer (pinned_array< T, Flags > &hostImgBuf)`
- `CUBOTH T * get (int i)`
- `CUBOTH T pixel_oobcheck (int x, int y, int imgIndex, T border=0.0f)`
- `CUBOTH T & pixel (int x, int y, int imgIndex)`
- `CUBOTH T * pixelAddress (int x, int y, int imgIndex)`
- `CUBOTH bool boundaryHit (float2 center, float radius)`
- `void free ()`
- `void copyImageToHost (int img, T *dst, bool async=false, cudaStream_t s=0)`
- `void copyImageToDevice (int img, T *src, bool async=false, cudaStream_t s=0)`
- `void copyToHost (T *dst, bool async=false, cudaStream_t s=0)`
- `void copyToDevice (T *src, bool async=false, cudaStream_t s=0)`
- `void copyToDevice (T *src, int numImages, bool async=false, cudaStream_t s=0)`
- `void clear ()`
- `CUBOTH int totalNumPixels ()`
- `CUBOTH int totalNumBytes ()`
- `CUBOTH T interpolate (float x, float y, int idx, bool &outside)`
- `void bind (texture< T, cudaTextureType2D, cudaReadModeElementType > &texref)`
- `void unbind (texture< T, cudaTextureType2D, cudaReadModeElementType > &texref)`
- `CUBOTH void computelImagePos (int &x, int &y, int idx)`
- `__device__ T interpolateFromTexture (texture< T, cudaTextureType2D, cudaReadModeElementType > texref, float x, float y, int idx, bool &outside)`

Static Public Member Functions

- static `cudalmageList< T > emptyList ()`
- static `cudalmageList< T > alloc (int w, int h, int amount)`
- static `CUBOTH T interp (T a, T b, float x)`

Public Attributes

- `T * data`
- `size_t pitch`
- `int w`
- `int h`
- `int count`

10.12.1 Member Enumeration Documentation

10.12.1.1 template<typename T> anonymous enum

Enumerator

MaxImageWidth

```
20 { MaxImageWidth = 8192 };
```

10.12.2 Member Function Documentation

10.12.2.1 template<typename T> static cudalmageList<T> cudalmageList< T >::alloc (int *w*, int *h*, int *amount*) [inline], [static]

```

36
37     cudaImageList imgl;
38     imgl.w = w; imgl.h = h;
39     imgl.count = amount;
40
41     if (cudaMallocPitch(&imgl.data, &imgl.pitch, sizeof(T)*imgl.
42         fullwidth(), imgl.fullheight()) != cudaSuccess) {
43         throw std::bad_alloc(Sprintf("cudaImageListf<%s> alloc %dx%dx%d failed", typeid(T).name(
44 ), w, h, amount).c_str());
45     }
46     return imgl;
47 }
```

10.12.2.2 template<typename T> template<int Flags> void cudalmageList< T >::allocateHostImageBuffer (pinned_array< T, Flags > & hostImgBuf) [inline]

```

48
49     hostImgBuf.init( numpixels() );
50 }
```

**10.12.2.3 template<typename T> void cudalmageList< T >::bind (texture< T, cudaTextureType2D,
cudaReadModeElementType > & texref) [inline]**

```

166
167     cudaChannelFormatDesc desc = cudaCreateChannelDesc<T>();
168     cudaBindTexture2D(NULL, &texref, data, &desc, w, h*count,
169     pitch);
170 }
```

10.12.2.4 template<typename T> CUBOTH bool cudalmageList< T >::boundaryHit (float2 center, float radius) [inline]

```

80     {
81         return center.x + radius >= w ||
82             center.x - radius < 0 ||
83             center.y + radius >= h ||
84             center.y - radius < 0;
85     }
```

10.12.2.5 template<typename T> CUBOTH int cudalmageList< T >::capacity () [inline]

```
22 { return count; }
```

10.12.2.6 template<typename T> void cudalmageList< T >::clear () [inline]

```

136
137     if(data) cudaMemset2D(data, pitch, 0, w*sizeof(T), count*
138     h);
139 }
```

10.12.2.7 template<typename T> CUBOTH void cudaImageList< T >::computeImagePos (int & x, int & y, int idx) [inline]

```
175     {
176         y += idx * h;
177     }
```

10.12.2.8 template<typename T> void cudaImageList< T >::copyImageToDevice (int img, T * src, bool async = false, cudaStream_t s = 0) [inline]

```
106     T* dst = pixelAddress (0,0, img);
107
108     if (async)
109         cudaMemcpy2DAsync(dst, pitch, src, w*sizeof(T), w*sizeof(T),
110                           h, cudaMemcpyHostToDevice, s);
111     else
112         cudaMemcpy2D(dst, pitch, src, w*sizeof(T), w*sizeof(T), h, cudaMemcpyHostToDevice);
113 }
```

10.12.2.9 template<typename T> void cudaImageList< T >::copyImageToHost (int img, T * dst, bool async = false, cudaStream_t s = 0) [inline]

```
97
98     T* src = pixelAddress (0,0, img);
99
100    if (async)
101        cudaMemcpy2DAsync(dst, sizeof(T)*w, src, pitch, w*sizeof(T), h, cudaMemcpyDeviceToHost,
102                           s);
103    else
104        cudaMemcpy2D(dst, sizeof(T)*w, src, pitch, w*sizeof(T), h, cudaMemcpyDeviceToHost);
```

10.12.2.10 template<typename T> void cudaImageList< T >::copyToDevice (T * src, bool async = false, cudaStream_t s = 0) [inline]

```
122
123     if (async)
124         cudaMemcpy2DAsync(data, pitch, src, w*sizeof(T), w*sizeof(T),
125                           count*h, cudaMemcpyHostToDevice, s);
126     else
127         cudaMemcpy2D(data, pitch, src, w*sizeof(T), w*sizeof(T),
128                           count*h, cudaMemcpyHostToDevice);
```

10.12.2.11 template<typename T> void cudaImageList< T >::copyToDevice (T * src, int numImages, bool async = false, cudaStream_t s = 0) [inline]

```
129
130     if (async)
131         cudaMemcpy2DAsync(data, pitch, src, w*sizeof(T), w*sizeof(T), numImages*
132                           h, cudaMemcpyHostToDevice, s);
133     else
134         cudaMemcpy2D(data, pitch, src, w*sizeof(T), w*sizeof(T), numImages*
135                           h, cudaMemcpyHostToDevice);
```

10.12.2.12 template<typename T> void cudalmageList< T >::copyToHost (T * dst, bool *async* = false, cudaStream_t *s* = 0) [inline]

```

115
116      if (async)
117          cudaMemcpy2DAsync(dst, sizeof(T)*w, data, pitch, w*sizeof(T),
118          count*h, cudaMemcpyDeviceToHost, s);
119      else
120          cudaMemcpy2D(dst, sizeof(T)*w, data, pitch, w*sizeof(T),
121          count*h, cudaMemcpyDeviceToHost);
120      }

```

10.12.2.13 template<typename T> static cudalmageList<T> cudalmageList< T >::emptyList () [inline], [static]

```

25
26     cudaImageList imgl;
27     imgl.data = 0;
28     imgl.pitch = 0;
29     imgl.w = imgl.h = 0;
30     imgl.count = 0;
31     return imgl;
32 }

```

10.12.2.14 template<typename T> void cudalmageList< T >::free () [inline]

```

89
90     {
91         if(data) {
92             cudaFree(data);
93             data=0;
94         }
94     }

```

10.12.2.15 template<typename T> CUBOTH int cudalmageList< T >::fullheight () [inline]

```
18 { return h*count; }
```

10.12.2.16 template<typename T> CUBOTH int cudalmageList< T >::fullwidth () [inline]

```
17 { return w; }
```

10.12.2.17 template<typename T> CUBOTH T* cudalmageList< T >::get (int *i*) [inline]

```

52
53     {
53     return (T*)((char*)data) + pitch*h*i;
54     }

```

10.12.2.18 template<typename T> static CUBOTH T cudalmageList< T >::interp (T *a*, T *b*, float *x*) [inline], [static]

```
143 { return a + (b-a)*x; }
```

10.12.2.19 template<typename T> CUBOTH T cudalmageList< T >::interpolate (float x, float y, int idx, bool & outside) [inline]

```

146     {
147         int rx=x, ry=y;
148
149         if (rx < 0 || ry < 0 || rx >= w-1 || ry >= h-1) {
150             outside=true;
151             return 0.0f;
152         }
153
154         T v00 = pixel(rx, ry, idx);
155         T v10 = pixel(rx+1, ry, idx);
156         T v01 = pixel(rx, ry+1, idx);
157         T v11 = pixel(rx+1, ry+1, idx);
158
159         T v0 = interp (v00, v10, x-rx);
160         T v1 = interp (v01, v11, x-rx);
161
162         outside=false;
163         return interp (v0, v1, y-ry);
164     }

```

10.12.2.20 template<typename T> __device__ T cudalmageList< T >::interpolateFromTexture (texture< T, cudaTextureType2D, cudaMemcpyModeType > texref, float x, float y, int idx, bool & outside) [inline]

```

181     {
182         int rx=x, ry=y;
183
184         if (rx < 0 || ry < 0 || rx >= w-1 || ry >= h-1) {
185             outside=true;
186             return 0.0f;
187         }
188
189         computeImagePos(rx, ry, idx);
190
191         float fx=x-floor(x), fy = y-floor(y);
192         float u = rx + 0.5f;
193         float v = ry + 0.5f;
194
195         T v00 = tex2D(texref, u, v);
196         T v10 = tex2D(texref, u+1, v);
197         T v01 = tex2D(texref, u, v+1);
198         T v11 = tex2D(texref, u+1, v+1);
199
200         T v0 = interp (v00, v10, fx);
201         T v1 = interp (v01, v11, fx);
202
203         outside = false;
204         return interp (v0, v1, fy);
205     }

```

10.12.2.21 template<typename T> CUBOTH bool cudalmageList< T >::isEmpty () [inline]

```
34 { return data==0; }
```

10.12.2.22 template<typename T> CUBOTH int cudalmageList< T >::numpixels () [inline]

```
23 { return w*h*count; }
```

10.12.2.23 template<typename T> CUBOTH T& cudalmageList< T >::pixel (int x, int y, int imgIndex) [inline]

```

65
66         computeImagePos(x,y,imgIndex);
67         T* row = (T*) ( (char*)data + y*pitch );
68         return row[x];
69     }

```

10.12.2.24 template<typename T> CUBOTH T cudalmageList< T >::pixel_oobcheck (int x, int y, int imgIndex, T border = 0.0f) [inline]

```

56
57     if (x < 0 || x >= w || y < 0 || y >= h)
58         return border;
59
60     computeImagePos(x,y,imgIndex);
61     T* row = (T*) ( (char*)data + y*pitch );
62     return row[x];
63 }
```

10.12.2.25 template<typename T> CUBOTH T* cudalmageList< T >::pixelAddress (int x, int y, int imgIndex) [inline]

```

71
72     computeImagePos(x,y,imgIndex);
73     T* row = (T*) ( (char*)data + y*pitch );
74     return row + x;
75 }
```

10.12.2.26 template<typename T> CUBOTH int cudalmageList< T >::totalNumBytes () [inline]

```
141 { return w*h*count*sizeof(T); }
```

10.12.2.27 template<typename T> CUBOTH int cudalmageList< T >::totalNumPixels () [inline]

```
140 { return w*h*count; }
```

10.12.2.28 template<typename T> void cudalmageList< T >::unbind (texture< T, cudaTextureType2D, cudaReadModeElementType > &texref) [inline]

```

170
171     cudaUnbindTexture(&texref);
172 }
```

10.12.3 Member Data Documentation

10.12.3.1 template<typename T> int cudalmageList< T >::count

10.12.3.2 template<typename T> T* cudalmageList< T >::data

10.12.3.3 template<typename T> int cudalmageList< T >::h

10.12.3.4 template<typename T> size_t cudalmageList< T >::pitch

10.12.3.5 template<typename T> int cudalmageList< T >::w

The documentation for this struct was generated from the following file:

- cudatrace/cudalmageList.h

10.13 QueuedCUDATracker::Device Struct Reference

```
#include <QueuedCUDATracker.h>
```

Public Member Functions

- `Device (int index)`
- `~Device ()`
- `void SetRadialZLUT (float *data, int radialsteps, int planes, int numLUTs)`
- `void SetPixelCalibrationImages (float *offset, float *gain, int img_width, int img_height)`
- `void SetRadialWeights (float *zcmp)`

Public Attributes

- `cudaImageListf radial_zlut`
- `cudaImageListf calib_offset`
- `cudaImageListf calib_gain`
- `device_vec< float > zcompareWindow`
- `QI::DeviceInstance qi_instance`
- `QI::DeviceInstance qalign_instance`
- `device_vec< float2 > zlut_trigtable`
- `int index`

10.13.1 Constructor & Destructor Documentation

10.13.1.1 QueuedCUDATracker::Device::Device (int *index*) [inline]

```
109
110         {
111             this->index=index;
112             radial_zlut=calib_offset=calib_gain=
113             cudaImageListf::emptyList();
114         }
```

10.13.1.2 QueuedCUDATracker::Device::~Device ()

```
229 {
230     cudaSetDevice(index);
231     radial_zlut.free();
232     calib_gain.free();
233     calib_offset.free();
234 }
```

10.13.2 Member Function Documentation

10.13.2.1 void QueuedCUDATracker::Device::SetPixelCalibrationImages (float * offset, float * gain, int img_width, int img_height)

```

786 {
787     cudaSetDevice(index);
788
789     if (offset == 0)
790         calib_offset.free();
791
792     if (gain == 0)
793         calib_gain.free();
794
795     if (radial_zlut.count > 0) {
796
797         if (gain)
798             calib_gain = cudaImageListf::alloc(img_width, img_height,
799             radial_zlut.count);
800
801         if (offset)
802             calib_offset = cudaImageListf::alloc(img_width, img_height,
803             radial_zlut.count);
804
805         for (int j=0;j<radial_zlut.count;j++) {
806             if (gain) calib_gain.copyImageToDevice(j, &gain[img_width*img_height
807             *j]);
808             if (offset) calib_offset.copyImageToDevice(j, &offset[img_width*
809             img_height*j]);
810         }
811     }
812 }
```

10.13.2.2 void QueuedCUDATracker::Device::SetRadialWeights (float * zcmp)

```

850 {
851     cudaSetDevice(index);
852     if (zcmp)
853         zcompareWindow.copyToDevice(zcmp, radial_zlut.
854         w, false);
855     else
856         zcompareWindow.free();
```

10.13.2.3 void QueuedCUDATracker::Device::SetRadialZLUT (float * data, int radialsteps, int planes, int numLUTs)

```

837 {
838     cudaSetDevice(index);
839
840     radial_zlut.free();
841     radial_zlut = cudaImageListf::alloc(radialsteps, planes, numLUTs);
842     if (data) {
843         for (int i=0;i<numLUTs;i++)
844             radial_zlut.copyImageToDevice(i, &data[planes*radialsteps*i]);
845     }
846     else radial_zlut.clear();
847 }
```

10.13.3 Member Data Documentation

10.13.3.1 cudalImageListf QueuedCUDATracker::Device::calib_gain

10.13.3.2 cudalImageListf QueuedCUDATracker::Device::calib_offset

10.13.3.3 int QueuedCUDATracker::Device::index

10.13.3.4 QI::DeviceInstance QueuedCUDATracker::Device::qalign_instance

10.13.3.5 QI::DeviceInstance QueuedCUDATracker::Device::qi_instance

10.13.3.6 cudalImageListf QueuedCUDATracker::Device::radial_zlut

10.13.3.7 device_vec<float> QueuedCUDATracker::Device::zcompareWindow

10.13.3.8 device_vec<float2> QueuedCUDATracker::Device::zlut_trigtable

The documentation for this struct was generated from the following files:

- cudatrack/[QueuedCUDATracker.h](#)
- cudatrack/[QueuedCUDATracker.cu](#)

10.14 device_vec< T > Class Template Reference

```
#include <gpu_utils.h>
```

Public Member Functions

- [device_vec \(\)](#)
- [device_vec \(size_t N\)](#)
- [device_vec \(const device_vec< T > &src\)](#)
- [device_vec \(const std::vector< T > &src\)](#)
- [~device_vec \(\)](#)
- void [init \(size_t s\)](#)
- void [free \(\)](#)
- [operator std::vector< T > \(\) const](#)
- [device_vec< T > & operator= \(const std::vector< T > &src\)](#)
- [device_vec< T > & operator= \(const device_vec< T > &src\)](#)
- void [copyToHost \(T *dst, bool async, cudaStream_t s=0\)](#)
- void [copyToHost \(std::vector< T > &dst, bool async, cudaStream_t s=0\)](#)
- void [copyToDevice \(const std::vector< T > &src, bool async=false, cudaStream_t s=0\)](#)
- void [copyToDevice \(const T *first, size_t size, bool async=false, cudaStream_t s=0\)](#)
- std::vector< T > [toVector \(\)](#)
- size_t [memsize \(\)](#)

Public Attributes

- size_t [size](#)
- T * [data](#)

10.14.1 Constructor & Destructor Documentation

10.14.1.1 template<typename T> device_vec< T >::device_vec() [inline]

```
68         {
69             data = 0;
70             size = 0;
71         }
```

10.14.1.2 template<typename T> device_vec< T >::device_vec(size_t N) [inline]

```
73         {
74             data = 0;
75             size = 0;
76             init(N);
77         }
```

10.14.1.3 template<typename T> device_vec< T >::device_vec(const device_vec< T > & src) [inline]

```
78         {
79             data = 0; size = 0;
80             init(src.size());
81             dbgCUDAErrorCheck(cudaMemcpy(data, src.data, sizeof(T) *
82                 size, cudaMemcpyDeviceToDevice));
83         }
```

10.14.1.4 template<typename T> device_vec< T >::device_vec(const std::vector< T > & src) [inline]

```
83         {
84             data=0; size=0;
85             init(src.size());
86             dbgCUDAErrorCheck(cudaMemcpy(data, &src[0], sizeof(T) *
87                 size, cudaMemcpyHostToDevice));
88         }
```

10.14.1.5 template<typename T> device_vec< T >::~device_vec() [inline]

```
88         {
89             free();
90         }
```

10.14.2 Member Function Documentation

10.14.2.1 template<typename T> void device_vec< T >::copyToDevice(const std::vector< T > & src, bool async = false, cudaStream_t s = 0) [inline]

```
135         {
136             copyToDevice(&src[0], src.size(), async, s);
137         }
```

10.14.2.2 template<typename T> void device_vec< T >::copyToDevice (const T * *first*, size_t *size*, bool *async* = false, cudaStream_t *s* = 0) [inline]

```
138     if (this->size < size)
139         init(size);
140     if (async)
141         dbgCUDAErrorCheck(cudaMemcpyAsync(data, first, sizeof(T) *
142             size, cudaMemcpyHostToDevice, s));
143     else
144         dbgCUDAErrorCheck(cudaMemcpy(data, first, sizeof(T) *
145             size, cudaMemcpyHostToDevice));
```

10.14.2.3 template<typename T> void device_vec< T >::copyToHost (T * *dst*, bool *async*, cudaStream_t *s* = 0) [inline]

```
124     if (async)
125         dbgCUDAErrorCheck(cudaMemcpyAsync(dst, data, sizeof(T) *
126             size, cudaMemcpyDeviceToHost, s));
127     else
128         dbgCUDAErrorCheck(cudaMemcpy(dst, data, sizeof(T) *
129             size, cudaMemcpyDeviceToHost));
```

10.14.2.4 template<typename T> void device_vec< T >::copyToHost (std::vector< T > & *dst*, bool *async*, cudaStream_t *s* = 0) [inline]

```
130     if (dst.size() != size)
131         dst.resize(size);
132     copyToHost(&dst[0], async, s);
```

10.14.2.5 template<typename T> void device_vec< T >::free () [inline]

```
102     {
103     if (data) {
104         dbgCUDAErrorCheck(cudaFree(data));
105         data=0;
106     }
107 }
```

10.14.2.6 template<typename T> void device_vec< T >::init (size_t *s*) [inline]

```
91     if(size != s) {
92         free();
93     }
94     if (s!=0) {
95         if (cudaMalloc(&data, sizeof(T)*s) != cudaSuccess) {
96             throw std::bad_alloc(Sprintf("device_vec<%s> init %d elements failed", typeid(T).
97                 name(), s).c_str());
98         }
99         size = s;
100    }
101 }
```

10.14.2.7 template<typename T> size_t device_vec< T >::memsize() [inline]

```
152 { return size*sizeof(T); }
```

10.14.2.8 template<typename T> device_vec< T >::operator std::vector< T >() const [inline]

```
108     std::vector<T> dst(size);
109     dbgCUDAErrorCheck(cudaMemcpy(&dst[0], data, sizeof(T)*
110     size, cudaMemcpyDeviceToHost));
111     return dst;
112 }
```

10.14.2.9 template<typename T> device_vec<T>& device_vec< T >::operator=(const std::vector< T > & src) [inline]

```
113     {
114         init(src.size());
115         dbgCUDAErrorCheck(cudaMemcpy(data, &src[0], sizeof(T)*
116         size, cudaMemcpyHostToDevice));
117         return *this;
118     }
```

10.14.2.10 template<typename T> device_vec<T>& device_vec< T >::operator=(const device_vec< T > & src) [inline]

```
118     {
119         clear();
120         init(src.size());
121         dbgCUDAErrorCheck(cudaMemcpy(data, src.data, sizeof(T)*
122         size, cudaMemcpyDeviceToDevice));
123         return *this;
124     }
```

10.14.2.11 template<typename T> std::vector<T> device_vec< T >::toVector() [inline]

```
147     {
148         std::vector<T> v (size);
149         dbgCUDAErrorCheck(cudaMemcpy(&v[0], data, sizeof(T)*
150         size, cudaMemcpyDeviceToHost));
151         return v;
152     }
```

10.14.3 Member Data Documentation

10.14.3.1 template<typename T> T* device_vec< T >::data

10.14.3.2 template<typename T> size_t device_vec< T >::size

The documentation for this class was generated from the following file:

- [cudatrack/gpu_utils.h](#)

10.15 QI::DeviceInstance Struct Reference

```
#include <QI.h>
```

Public Member Functions

- [~DeviceInstance \(\)](#)
- [DeviceInstance \(\)](#)

Public Attributes

- [device_vec< float2 > qi_trigtable](#)
- [QIParams * d_qiparams](#)
- [device_vec< float > d_radialweights](#)

10.15.1 Constructor & Destructor Documentation

10.15.1.1 QI::DeviceInstance::~DeviceInstance () [inline]

```
54 { cudaFree(d_qiparams); }
```

10.15.1.2 QI::DeviceInstance::DeviceInstance () [inline]

```
55 { d_qiparams=0; }
```

10.15.2 Member Data Documentation

10.15.2.1 QIParams* QI::DeviceInstance::d_qiparams

10.15.2.2 device_vec<float> QI::DeviceInstance::d_radialweights

10.15.2.3 device_vec<float2> QI::DeviceInstance::qi_trigtable

The documentation for this struct was generated from the following file:

- cudatrack/QI.h

10.16 ErrorCluster Struct Reference

```
#include <labview.h>
```

Public Attributes

- LVBoolean `status`
- int32 `code`
- LStrHandle `message`

10.16.1 Member Data Documentation

10.16.1.1 int32 ErrorCluster::code

10.16.1.2 LStrHandle ErrorCluster::message

10.16.1.3 LVBoolean ErrorCluster::status

The documentation for this struct was generated from the following file:

- cputrack/labview.h

10.17 CPUTracker::FFT2D Class Reference

```
#include <cpu_tracker.h>
```

Public Member Functions

- `FFT2D (int w, int h)`
- `~FFT2D ()`
- void `Apply (float *d)`

Public Attributes

- `kissfft< float > xfft`
- `kissfft< float > yfft`
- `std::complex< float > * cbuf`

10.17.1 Constructor & Destructor Documentation

10.17.1.1 CPUTracker::FFT2D::FFT2D (int w, int h) [inline]

```
58 : xfft(w, false), yfft(h, false) { cbuf=new std::complex<float>[w*h]; }
```

10.17.1.2 CPUTracker::FFT2D::~FFT2D() [inline]

```
59 { delete[] cbuf; }
```

10.17.2 Member Function Documentation

10.17.2.1 void CPUTracker::FFT2D::Apply (float * d)

```

1056 {
1057     int w = xfft.nfft();
1058     int h = yfft.nfft();
1059
1060     std::complex<float>* tmpsrc = ALLOCA_ARRAY(std::complex<float>, h);
1061     std::complex<float>* tmpdst = ALLOCA_ARRAY(std::complex<float>, h);
1062
1063     for(int y=0;y<h;y++) {
1064         for (int x=0;x<w;x++)
1065             tmpsrc[x]=d[y*w+x];
1066         xfft.transform(tmpsrc, &cbuf[y*w]);
1067     }
1068     for (int x=0;x<w;x++) {
1069         for (int y=0;y<h;y++)
1070             tmpsrc[y]=cbuf[y*w+x];
1071         yfft.transform(tmpsrc, tmpdst);
1072         for (int y=0;y<h;y++)
1073             cbuf[y*w+x]=tmpdst[y];
1074     }
1075     // copy and shift
1076     for (int y=0;y<h;y++) {
1077         for (int x=0;x<w;x++) {
1078             int dx=(x+w/2)%w;
1079             int dy=(y+h/2)%h;
1080             auto v=cbuf[x+y*w];
1081             d[dx+dy*w] = v.real()*v.real() + v.imag()*v.imag();
1082         }
1083     }
1084
1085 // delete[] tmpsrc;
1086 // delete[] tmpdst;
1087 }
```

10.17.3 Member Data Documentation

10.17.3.1 std::complex<float>* CPUTracker::FFT2D::cbuf

10.17.3.2 kissfft<float> CPUTracker::FFT2D::xfft

10.17.3.3 kissfft<float> CPUTracker::FFT2D::yfft

The documentation for this class was generated from the following files:

- cputrack/cpu_tracker.h
- cputrack/cpu_tracker.cpp

10.18 FFT2DTracker Class Reference

```
#include <FFT2DTracker.h>
```

Public Member Functions

- [FFT2DTracker \(int w, int h\)](#)
- [~FFT2DTracker \(\)](#)
- [vector2f ComputeXCor \(float *image\)](#)
- [float * GetAutoConvResults \(\)](#)
- [vector2f ComputeMax2DInterpolated \(float *img\)](#)

Public Attributes

- int `width`
- int `height`
- `fftw_plan_t plan_fw2D`
- `fftw_plan_t plan_bw2D`
- float * `mirror2D`
- complexc * `fft_buf`
- complexc * `fft_buf_mirrored`

10.18.1 Constructor & Destructor Documentation

10.18.1.1 `FFT2DTracker::FFT2DTracker(int w, int h)`

10.18.1.2 `FFT2DTracker::~FFT2DTracker()`

10.18.2 Member Function Documentation

10.18.2.1 `vector2f FFT2DTracker::ComputeMax2DInterpolated(float * img)`

10.18.2.2 `vector2f FFT2DTracker::ComputeXCor(float * image)`

10.18.2.3 `float* FFT2DTracker::GetAutoConvResults() [inline]`

```
14 { return mirror2D; }
```

10.18.3 Member Data Documentation

10.18.3.1 `complexc* FFT2DTracker::fft_buf`

10.18.3.2 `complexc * FFT2DTracker::fft_buf_mirrored`

10.18.3.3 `int FFT2DTracker::height`

10.18.3.4 `float* FFT2DTracker::mirror2D`

10.18.3.5 `fftw_plan_t FFT2DTracker::plan_bw2D`

10.18.3.6 `fftw_plan_t FFT2DTracker::plan_fw2D`

10.18.3.7 `int FFT2DTracker::width`

The documentation for this class was generated from the following file:

- cputrack/[FFT2DTracker.h](#)

10.19 ResultManager::FrameCounters Struct Reference

```
#include <ResultManager.h>
```

Public Member Functions

- [FrameCounters \(\)](#)

Public Attributes

- int [startFrame](#)
- int [processedFrames](#)
- int [lastSaveFrame](#)
- int [capturedFrames](#)
- int [localizationsDone](#)
- int [lostFrames](#)
- int [fileError](#)

10.19.1 Constructor & Destructor Documentation

10.19.1.1 ResultManager::FrameCounters()

```
32 {  
33     startFrame = 0;  
34     lastSaveFrame = 0;  
35     processedFrames = 0;  
36     capturedFrames = 0;  
37     localizationsDone = 0;  
38     lostFrames = 0;  
39     fileError = 0;  
40 }
```

10.19.2 Member Data Documentation

10.19.2.1 int ResultManager::FrameCounters::capturedFrames

10.19.2.2 int ResultManager::FrameCounters::fileError

10.19.2.3 int ResultManager::FrameCounters::lastSaveFrame

10.19.2.4 int ResultManager::FrameCounters::localizationsDone

10.19.2.5 int ResultManager::FrameCounters::lostFrames

10.19.2.6 int ResultManager::FrameCounters::processedFrames

10.19.2.7 int ResultManager::FrameCounters::startFrame

The documentation for this struct was generated from the following files:

- cputrack/[ResultManager.h](#)
- cputrack/[ResultManager.cpp](#)

10.20 ResultManager::FrameResult Struct Reference

```
#include <ResultManager.h>
```

Public Member Functions

- [FrameResult](#) (int nResult, int nFrameInfo)

Public Attributes

- std::vector< [LocalizationResult](#) > results
- std::vector< float > frameInfo
- int count
- double timestamp
- bool hasFrameInfo

10.20.1 Constructor & Destructor Documentation

10.20.1.1 ResultManager::FrameResult (int nResult, int nFrameInfo) [inline]

```
100 : frameInfo(nFrameInfo), results(nResult) { count=0;
    timestamp=0; hasFrameInfo=false; }
```

10.20.2 Member Data Documentation

10.20.2.1 int ResultManager::FrameResult::count

10.20.2.2 std::vector<float> ResultManager::FrameResult::frameInfo

10.20.2.3 bool ResultManager::FrameResult::hasFrameInfo

10.20.2.4 std::vector<LocalizationResult> ResultManager::FrameResult::results

10.20.2.5 double ResultManager::FrameResult::timestamp

The documentation for this struct was generated from the following file:

- cputrack/[ResultManager.h](#)

10.21 CPUTracker::Gauss2DResult Struct Reference

```
#include <cpu_tracker.h>
```

Public Attributes

- `vector2f pos`
- `float I0`
- `float bg`

10.21.1 Member Data Documentation

10.21.1.1 `float CPUTracker::Gauss2DResult::bg`

10.21.1.2 `float CPUTracker::Gauss2DResult::I0`

10.21.1.3 `vector2f CPUTracker::Gauss2DResult::pos`

The documentation for this struct was generated from the following file:

- `cputrack/cpu_tracker.h`

10.22 Threads::Handle Struct Reference

```
#include <threads.h>
```

Public Attributes

- `DWORD threadID`
- `ThreadEntryPoint callback`
- `HANDLE winhdl`
- `void * param`

10.22.1 Member Data Documentation

10.22.1.1 `ThreadEntryPoint Threads::Handle::callback`

10.22.1.2 `void* Threads::Handle::param`

10.22.1.3 `DWORD Threads::Handle::threadID`

10.22.1.4 `HANDLE Threads::Handle::winhdl`

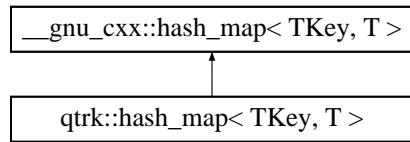
The documentation for this struct was generated from the following file:

- `cputrack/threads.h`

10.23 qtrk::hash_map< TKey, T > Class Template Reference

```
#include <hash_templates.h>
```

Inheritance diagram for qtrk::hash_map< TKey, T >:



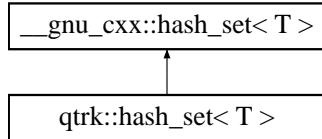
The documentation for this class was generated from the following file:

- cputrack/hash_templates.h

10.24 qtrk::hash_set< T > Class Template Reference

```
#include <hash_templates.h>
```

Inheritance diagram for qtrk::hash_set< T >:



The documentation for this class was generated from the following file:

- cputrack/hash_templates.h

10.25 Image4DCudaArray< T > Struct Template Reference

```
#include <cudaImageList.h>
```

Classes

- struct [KernelInst](#)

Public Member Functions

- `cudaExtent getExtent ()`
- `KernellInst kernellInst ()`
- `void bind (texture< T, cudaTextureType2DLayered, cudaMemcpyModeElementType > &texref)`
- `void unbind (texture< T, cudaTextureType2DLayered, cudaMemcpyModeElementType > &texref)`
- `void bind (surface< void, cudaSurfaceType2DLayered > &surf)`
- `Image4DCudaArray (int sx, int sy, int numImg, int sL)`
- `int2 getImagePos (int image)`
- `~Image4DCudaArray ()`
- `void copyToDevice (T *src, bool async=false, cudaStream_t s=0)`
- `void copyToHost (T *dst, bool async=false, cudaStream_t s=0)`
- `void clear ()`
- `void copyImageToHost (int img, int layer, T *dst, bool async=false, cudaStream_t s=0)`
- `void copyImageToDevice (int img, int layer, T *src, bool async=false, cudaStream_t s=0)`
- `void free ()`

Public Attributes

- `cudaArray_t array`
- `int imgw`
- `int imgh`
- `int layerw`
- `int layerh`
- `int nlayers`
- `int numImg`

10.25.1 Constructor & Destructor Documentation

10.25.1.1 template<typename T> Image4DCudaArray< T >::Image4DCudaArray (int sx, int sy, int numImg, int sL) [inline]

```

271
272     array = 0;
273     int d;
274     cudaGetDevice(&d);
275     cudaDeviceProp prop;
276     cudaGetDeviceProperties(&prop, d);
277
278     imgw = sx;
279     imgh = sy;
280     this->numImg = numImg;
281
282 //    layerh = (int)(prop.maxSurface2DLayered[1] / imgh);
283 //    layerh = 2048 / imgh;
284     layerw = (numImg + layerh - 1) / layerh;
285     nlayers = sL;
286
287     dbgprintf("creating image4D: %d layers of %d x %d images of %d x %d (%dx%dx%d)\n",
288             sL, layerw, layerh, imgw, imgh, getExtent().width,
289             getExtent().height, getExtent().depth);
290
291     cudaChannelFormatDesc desc = cudaCreateChannelDesc< T >();
292     cudaError_t err = cudaMalloc3DArray(&array, &desc, getExtent(), cudaArrayLayered |
293                                         cudaArraySurfaceLoadStore);
294     //cudaError_t err = cudaMalloc3DArray(&array, &desc, getExtent(), cudaArraySurfaceLoadStore);
295     if (err != cudaSuccess) {
296         throw std::bad_alloc(SPrint("CUDA error during cudaSurf2DList(): %s",
297             cudaGetErrorString(err)).c_str());
298     }
299 }
```

10.25.1.2 template<typename T > Image4DCudaArray< T >::~Image4DCudaArray() [inline]

```
303             {
304         free();
305     }
```

10.25.2 Member Function Documentation

10.25.2.1 template<typename T > void Image4DCudaArray< T >::bind(texture< T, cudaTextureType2DLayered, cudaReadModeElementType > & texref) [inline]

```
256             {
257         cudaChannelFormatDesc desc = cudaCreateChannelDesc<T>();
258         CheckCUDAError( cudaBindTextureToArray(texref, array, &desc) );
259     }
```

10.25.2.2 template<typename T > void Image4DCudaArray< T >::bind(surface< void, cudaSurfaceType2DLayered > & surf) [inline]

```
264             {
265         cudaChannelFormatDesc desc = cudaCreateChannelDesc<T>();
266         CheckCUDAError( cudaBindSurfaceToArray(surf, array) );
267     }
```

10.25.2.3 template<typename T > void Image4DCudaArray< T >::clear() [inline]

```
324     {
325         // create a new black image in device memory and use to it clear all the layers
326         T* d;
327         size_t srcpitch;
328         CheckCUDAError( cudaMallocPitch(&d, &srcpitch, sizeof(T)*
329             imgw, imgh) );
330         CheckCUDAError( cudaMemset2D(d, srcpitch, 0, sizeof(T)*
331             imgw, imgh) );
332         cudaMemcpy3DParms p = {0};
333         p.dstArray = array;
334         p.extent = make_cudaExtent(imgw, imgh, 1);
335         p.kind = cudaMemcpyDeviceToDevice;
336         p.srcPtr = make_cudaPitchedPtr(d, srcpitch, sizeof(T)*imgw, imgh);
337         for (int l=0;l<nlayers;l++)
338             for (int img=0;img<numImg;img++) {
339                 int2 imgpos = getImagePos(img);
340                 p.dstPos.z = 1;
341                 p.dstPos.x = imgpos.x;
342                 p.dstPos.y = imgpos.y;
343                 CheckCUDAError( cudaMemcpy3D(&p) );
344             }
345         CheckCUDAError( cudaFree(d) );
346     }
```

10.25.2.4 template<typename T> void Image4DCudaArray< T >::copyImageToDevice (int img, int layer, T * src, bool *async* = false, cudaStream_t s = 0) [inline]

```

371     {
372         // Memcpy3D needs the right pitch for the source, so we first need to copy it to 2D pitched memory
373         // before moving the data to the cuda array
374         cudaMallocPitch(
375
376             cudaMemcpy3DParms p = {0};
377             p.dstArray = array;
378             int2 imgpos = getImagePos(img);
379
380             //The srcPos and dstPos fields are optional offsets into the source and destination objects and are
381             //defined in units of each object's elements.
382             // The element for a host or device pointer is assumed to be unsigned char. For CUDA arrays,
383             //positions must be in the range [0, 2048) for any dimension.
384             p.dstPos.z = layer;
385             p.dstPos.x = imgpos.x;
386             p.dstPos.y = imgpos.y;
387             p.extent = make_cudaExtent(imgw,imgh,1);
388             p.kind = cudaMemcpyHostToDevice;
389             p.srcPtr = make_cudaPitchedPtr(src, sizeof(T)*imgw, sizeof(T)*imgw,
390             imgh);
391             if (async)
392                 CheckCUDAError( cudaMemcpy3DAsync(&p, s) );
393             else
394                 CheckCUDAError( cudaMemcpy3D(&p) );
395     }

```

10.25.2.5 template<typename T> void Image4DCudaArray< T >::copyImageToHost (int img, int layer, T * dst, bool *async* = false, cudaStream_t s = 0) [inline]

```

349     {
350         // According to CUDA docs:
351         //      The extent field defines the dimensions of the transferred area in elements.
352         //      If a CUDA array is participating in the copy, the extent is defined in terms of that
353         //      array's elements.
354         //      If no CUDA array is participating in the copy then the extents are defined in elements of
355         //      unsigned char.
356         cudaMemcpy3DParms p = {0};
357         p.srcArray = array;
358         p.extent = make_cudaExtent(imgw,imgh,1);
359         p.kind = cudaMemcpyDeviceToHost;
360         p.srcPos.z = layer;
361         int2 imgpos = getImagePos(img);
362         p.srcPos.x = imgpos.x;
363         p.srcPos.y = imgpos.y;
364         p.dstPtr = make_cudaPitchedPtr(dst, sizeof(T)*imgw, sizeof(T)*imgw,
365         imgh);
366         if (async)
367             CheckCUDAError( cudaMemcpy3DAsync(&p, s) );
368         else
369             CheckCUDAError( cudaMemcpy3D(&p) );
370     }

```

10.25.2.6 template<typename T> void Image4DCudaArray< T >::copyToDevice (T * src, bool *async* = false, cudaStream_t s = 0) [inline]

```

308     {
309         for (int L=0;L<nlayers;L++) {
310             for (int i=0;i<numImg;i++)
311                 copyImageToDevice(i, L, &src[ imgw * imgh * ( numImg * L + i ) ],
312                 async, s);
313         }

```

10.25.2.7 template<typename T > void Image4DCudaArray< T >::copyToHost (T * dst, bool async = false, cudaStream_t s = 0) [inline]

```
316     {
317         for (int L=0;L<nlayers;L++) {
318             for (int i=0;i<numImg;i++)
319                 copyImageToHost(i, L, &dst[ imgw * imgb * ( numImg * L + i ) ], 
320                 async, s);
321     }
```

10.25.2.8 template<typename T > void Image4DCudaArray< T >::free () [inline]

```
393     {
394         if (array) {
395             CheckCUDAEError( cudaFreeArray(array) );
396             array = 0;
397         }
398     }
```

10.25.2.9 template<typename T > cudaExtent Image4DCudaArray< T >::getExtent () [inline]

```
222     {
223         return make_cudaExtent(imgw * layerw, imgb * layerh,
224             nlayers);
225     }
```

10.25.2.10 template<typename T > int2 Image4DCudaArray< T >::getImagePos (int image) [inline]

```
298     int2 pos = { imgw * ( image % layerw ), imgb * ( image / 
299         layerw ) };
300     return pos;
301 }
```

10.25.2.11 template<typename T > KernelInst Image4DCudaArray< T >::kernelInst () [inline]

```
249     {
250         KernelInst inst;
251         inst.imgw = imgw; inst.imgb = imgb;
252         inst.layerw = layerw;
253         return inst;
254     }
```

**10.25.2.12 template<typename T > void Image4DCudaArray< T >::unbind (texture< T, cudaTextureType2DLayered,
cudaReadModeElementType > & texref) [inline]**

```
260     {
261         cudaUnbindTexture(texref);
262     }
```

10.25.3 Member Data Documentation

10.25.3.1 template<typename T> cudaArray_t Image4DCudaArray< T >::array

10.25.3.2 template<typename T> int Image4DCudaArray< T >::imgh

10.25.3.3 template<typename T> int Image4DCudaArray< T >::imgw

10.25.3.4 template<typename T> int Image4DCudaArray< T >::layerh

10.25.3.5 template<typename T> int Image4DCudaArray< T >::layerw

10.25.3.6 template<typename T> int Image4DCudaArray< T >::nlayers

10.25.3.7 template<typename T> int Image4DCudaArray< T >::numImg

The documentation for this struct was generated from the following file:

- cudatrack/cudalImageList.h

10.26 Image4DMemory< T > Class Template Reference

```
#include <cudalImageList.h>
```

Classes

- struct [KernelParams](#)

Public Member Functions

- [Image4DMemory](#) (int w, int h, int d, int L)
- [~Image4DMemory](#) ()
- void [free](#) ()
- void [copyToDevice](#) (T *src, bool async=false, cudaStream_t s=0)
- void [copyToHost](#) (T *dst, bool async=false, cudaStream_t s=0)
- void [clear](#) ()
- float * [getImgAddr](#) (int2 imgpos)
- void [copyImageToHost](#) (int z, int l, T *dst, bool async=false, cudaStream_t s=0)
- void [copyImageToDevice](#) (int z, int l, T *src, bool async=false, cudaStream_t s=0)
- [KernelParams](#) [bind](#) ()
- void [unbind](#) ()

Static Public Member Functions

- static __device__ T [read](#) (const [KernelParams](#) &kp, int x, int y, int2 imgpos)
- static __device__ void [write](#) (T value, const [KernelParams](#) &kp, int x, int y, int2 imgpos)

Public Attributes

- KernelParams kp
- int layers
- int totalImg
- int rows

10.26.1 Constructor & Destructor Documentation

10.26.1.1 template<typename T> Image4DMemory< T >::Image4DMemory(int w, int h, int d, int L) [inline]

```

424
425     kp.imgh = h;
426     kp.imgw = w;
427     kp.depth = d;
428     layers = L;
429     totalImg = d*L;
430
431     rows = 2048 / kp.imgh;
432     kp.cols = (totalImg + rows - 1) / rows;
433
434     CheckCUDAError( cudaMallocPitch( &kp.d_data, &kp.
435     pitch, sizeof(T) * kp.cols * kp.imgw, rows * kp.imgh ) );
435 }
```

10.26.1.2 template<typename T> Image4DMemory< T >::~Image4DMemory() [inline]

```

437
438     free();
439 }
```

10.26.2 Member Function Documentation

10.26.2.1 template<typename T> KernelParams Image4DMemory< T >::bind() [inline]

```
494 { return kp; }
```

10.26.2.2 template<typename T> void Image4DMemory< T >::clear() [inline]

```

463
464     {
465         cudaMemset2D(kp.d_data, kp.pitch, 0, sizeof(T)*(kp.cols*
465         kp.imgw), rows*kp.imgh);
465     }
```

10.26.2.3 template<typename T> void Image4DMemory< T >::copyImageToDevice(int z, int l, T * src, bool async = false, cudaStream_t s = 0) [inline]

```

485
486     {
487         int2 imgpos = kp.GetImagePos(z, l);
488         if (async)
489             cudaMemcpy2DAsync(getImgAddr(imgpos), kp.pitch, src, sizeof(T)*
489             kp.imgw, sizeof(T)*kp.imgw, kp.imgh, cudaMemcpyHostToDevice, s);
490         else
491             cudaMemcpy2D(getImgAddr(imgpos), kp.pitch, src, sizeof(T)*
491             kp.imgw, sizeof(T)*kp.imgw, kp.imgh, cudaMemcpyHostToDevice);
491     }
```

10.26.2.4 template<typename T > void Image4DMemory< T >::copyImageToHost (int z, int l, T * dst, bool async = false, cudaStream_t s = 0) [inline]

```
476     {
477         int2 imgpos = kp.GetImagePos(z, l);
478         if (async)
479             cudaMemcpy2DAsync(dst, sizeof(T)*kp.imgw, getImgAddr(imgpos),
480                             kp.pitch, kp.imgw * sizeof(T), kp.imgh, cudaMemcpyDeviceToHost, s);
481         else
482             cudaMemcpy2D(dst, sizeof(T)*kp.imgw, getImgAddr(imgpos),
483                         kp.pitch, kp.imgw * sizeof(T), kp.imgh, cudaMemcpyDeviceToHost);
482 }
```

10.26.2.5 template<typename T > void Image4DMemory< T >::copyToDevice (T * src, bool async = false, cudaStream_t s = 0) [inline]

```
446     {
447         for (int L=0;L<layers;L++) {
448             for (int i=0;i<kp.depth;i++)
449                 copyImageToDevice(i, L, &src[ kp.imgw *
450 kp.imgh * ( kp.depth * L + i ) ], async, s);
451     }
```

10.26.2.6 template<typename T > void Image4DMemory< T >::copyToHost (T * dst, bool async = false, cudaStream_t s = 0) [inline]

```
454     {
455         for (int L=0;L<layers;L++) {
456             for (int i=0;i<kp.depth;i++)
457                 copyImageToHost(i, L, &dst[ kp.imgw * kp.
458 imgh * ( kp.depth * L + i ) ], async, s);
459     }
```

10.26.2.7 template<typename T > void Image4DMemory< T >::free() [inline]

```
440     {
441         if(kp.d_data) cudaFree(kp.d_data);
442         kp.d_data=0;
443     }
```

10.26.2.8 template<typename T > float* Image4DMemory< T >::getImgAddr (int2 imgpos) [inline]

```
468     {
469         char* d = (char*)kp.d_data;
470         d += imgpos.y * kp.pitch;
471         return &((float*)d)[imgpos.x];
472     }
```

10.26.2.9 template<typename T > static __device__ T Image4DMemory< T >::read (const KernelParams & kp, int x, int y, int2 imgpos) [inline], [static]

```
497     return ((T*)( (char*)kp.d_data + (y + imgpos.y) * kp.pitch))[ x + imgpos.x ];
498 }
499 }
```

10.26.2.10 template<typename T> void Image4DMemory< T >::unbind() [inline]

```
495 { }
```

10.26.2.11 template<typename T> static __device__ void Image4DMemory< T >::write(T value, const KernelParams & kp, int x, int y, int2 imgpos) [inline], [static]

```
500      ((T*) ( (char*) kp.d_data + (y + imgpos.y) * kp.pitch)) [ x + imgpos.x ] = value;
501
502 }
```

10.26.3 Member Data Documentation

10.26.3.1 template<typename T> KernelParams Image4DMemory< T >::kp

10.26.3.2 template<typename T> int Image4DMemory< T >::layers

10.26.3.3 template<typename T> int Image4DMemory< T >::rows

10.26.3.4 template<typename T> int Image4DMemory< T >::totalImg

The documentation for this class was generated from the following file:

- cudatrack/cudalImageList.h

10.27 ImageLUTConfig Struct Reference

```
#include <QueuedCUDATracker.h>
```

Static Public Member Functions

- static ImageLUTConfig empty()

Public Attributes

- int nLUTs
- int planes
- int w
- int h
- float xscale
- float yscale

10.27.1 Member Function Documentation

10.27.1.1 static ImageLUTConfig ImageLUTConfig::empty() [inline], [static]

```

53
54     ImageLUTConfig v;
55     v.nLUTs = v.planes=v.w=v.h=0;
56     v.xscale=v.yscale=1.0f;
57     return v;
58 }
```

10.27.2 Member Data Documentation

10.27.2.1 int ImageLUTConfig::h

10.27.2.2 int ImageLUTConfig::nLUTs

10.27.2.3 int ImageLUTConfig::planes

10.27.2.4 int ImageLUTConfig::w

10.27.2.5 float ImageLUTConfig::xscale

10.27.2.6 float ImageLUTConfig::yscale

The documentation for this struct was generated from the following file:

- cudatrack/QueuedCUDATracker.h

10.28 ImageSampler_InterpolatedTexture Class Reference

```
#include <ImageSampler.h>
```

Static Public Member Functions

- static void [BindTexture](#) (cudalImageListf &images)
- static void [UnbindTexture](#) (cudalImageListf &images)
- static __device__ float [Interpolated](#) (cudalImageListf &images, float x, float y, int img, bool &outside)
- static __device__ float [Index](#) (cudalImageListf &images, int x, int y, int img)

10.28.1 Member Function Documentation

10.28.1.1 static void ImageSampler_InterpolatedTexture::BindTexture (cudalImageListf & *images*) [inline], [static]

```
59 { images.bind(image_sampler_texture_nearest); }
```

```

10.28.1.2 static __device__ float ImageSampler_InterpolatedTexture::Index ( cudalImageListf & images, int x, int y, int img ) [inline], [static]

70      {
71          return tex2D(image_sampler_texture_nearest, x+0.5f,y+0.5f + img*images
72          .h);
    }

10.28.1.3 static __device__ float ImageSampler_InterpolatedTexture::Interpolated ( cudalImageListf & images, float x, float y, int img, bool & outside ) [inline], [static]

64      {
65          return images.interpolateFromTexture (
66          image_sampler_texture_nearest, x, y, img, outside);
    }

10.28.1.4 static void ImageSampler_InterpolatedTexture::UnbindTexture ( cudalImageListf & images ) [inline], [static]

60 { images.unbind(image_sampler_texture_nearest); }

```

The documentation for this class was generated from the following file:

- [cudatrack/ImageSampler.h](#)

10.29 ImageSampler_MemCopy Class Reference

```
#include <ImageSampler.h>
```

Static Public Member Functions

- [static void BindTexture \(cudalImageListf &images\)](#)
- [static void UnbindTexture \(cudalImageListf &images\)](#)
- [static __device__ float Interpolated \(cudalImageListf &images, float x, float y, int img, bool &outside\)](#)
- [static __device__ float Index \(cudalImageListf &images, int x, int y, int img\)](#)

10.29.1 Member Function Documentation

10.29.1.1 static void ImageSampler_MemCopy::BindTexture (cudalImageListf & images) [inline], [static]

```
6 { }
```

10.29.1.2 static __device__ float ImageSampler_MemCopy::Index (cudalImageListf & images, int x, int y, int img) [inline], [static]

```

17      {
18          return images.pixel(x, y, img);
19      }

```

10.29.1.3 static __device__ float ImageSampler_MemCopy::Interpolated (**cudalmageListf** & *images*, float *x*, float *y*, int *img*, bool & *outside*) [inline], [static]

```
11     {
12         return images.interpolate(x,y,img, outside);
13     }
```

10.29.1.4 static void ImageSampler_MemCopy::UnbindTexture (**cudalmageListf** & *images*) [inline], [static]

```
7 { }
```

The documentation for this class was generated from the following file:

- [cudatrack/ImageSampler.h](#)

10.30 ImageSampler_SimpleTextureRead Class Reference

```
#include <ImageSampler.h>
```

Static Public Member Functions

- static void [BindTexture \(cudalmageListf &images\)](#)
- static void [UnbindTexture \(cudalmageListf &images\)](#)
- static __device__ float [Interpolated \(cudalmageListf &images, float x, float y, int img, bool &outside\)](#)
- static __device__ float [Index \(cudalmageListf &images, int x, int y, int img\)](#)

Static Private Member Functions

- static __device__ float [ofs \(float x\)](#)

10.30.1 Member Function Documentation

10.30.1.1 static void ImageSampler_SimpleTextureRead::BindTexture (**cudalmageListf** & *images*) [inline], [static]

```
27 { images.bind(image_sampler_texture_linear); }
```

10.30.1.2 static __device__ float ImageSampler_SimpleTextureRead::Index (**cudalmageListf** & *images*, int *x*, int *y*, int *img*) [inline], [static]

```
44     {
45         return tex2D(image_sampler_texture_linear,
46         ofs(x),ofs(y) + img*images.h);
47     }
```

10.30.1.3 static __device__ float ImageSampler_SimpleTextureRead::Interpolated (`cudaImageListf & images`, `float x`, `float y`, `int img`, `bool & outside`) [inline], [static]

```
32      {
33          if (x < 0 || x >= images.w-1 || y < 0 || y >= images.h-1) {
34              outside=true;
35              return 0.0f;
36          } else {
37              outside=false;
38              return tex2D(image_sampler_texture_linear,
39                           ofs(x),ofs(y) + img*images.h);
40          }
41      }
```

10.30.1.4 static __device__ float ImageSampler_SimpleTextureRead::ofs (`float x`) [inline], [static], [private]

```
49 { return x+0.5f; }
```

10.30.1.5 static void ImageSampler_SimpleTextureRead::UnbindTexture (`cudaImageListf & images`) [inline], [static]

```
28 { images.unbind(image_sampler_texture_linear); }
```

The documentation for this class was generated from the following file:

- [cudatrack/ImageSampler.h](#)

10.31 QueuedCPUTracker::Job Struct Reference

Public Member Functions

- [Job \(\)](#)
- [~Job \(\)](#)

Public Attributes

- [uchar * data](#)
- [QTRK_PixelDataType dataType](#)
- [LocalizationJob job](#)

10.31.1 Constructor & Destructor Documentation

10.31.1.1 QueuedCPUTracker::Job::Job() [inline]

```
72 { data=0; dataType=QTrkU8; }
```

10.31.1.2 QueuedCPUTracker::Job::~Job() [inline]

```
73 { delete[] data; }
```

10.31.2 Member Data Documentation

10.31.2.1 uchar* QueuedCPUTracker::Job::data

10.31.2.2 QTRK_PixelDataType QueuedCPUTracker::Job::dataType

10.31.2.3 LocalizationJob QueuedCPUTracker::Job::job

The documentation for this struct was generated from the following file:

- cputrack/[QueuedCPUTracker.h](#)

10.32 Image4DCudaArray< T >::KernelInst Struct Reference

```
#include <cudaImageList.h>
```

Public Member Functions

- CUBOTH int2 [getImagePos](#) (int image)
- __device__ T [readSurfacePixel](#) (surface< void, cudaSurfaceType2DLayered > surf, int x, int y, int z)
- __device__ void [writeSurfacePixel](#) (surface< void, cudaSurfaceType2DLayered > surf, int x, int y, int z, T value)

Public Attributes

- int [imgw](#)
- int [imgh](#)
- int [layerw](#)

10.32.1 Member Function Documentation

10.32.1.1 template<typename T > CUBOTH int2 Image4DCudaArray< T >::KernelInst::getImagePos (int *image*) [inline]

```
232     {
233         return make_int2(imgw * (image % layerw), imgh * (image / layerw));
234     }
```

10.32.1.2 template<typename T> __device__ T Image4DCudaArray<T>::KernelInst::readSurfacePixel (surface< void, cudaSurfaceType2DLayered > surf, int x, int y, int z) [inline]

```
237     {
238         T r;
239         surf2DLayeredread (&r, image_lut_surface, sizeof(T)*x, y, z,
240         cudaBoundaryModeTrap);
240         return r;
241     }
```

10.32.1.3 template<typename T> __device__ void Image4DCudaArray<T>::KernelInst::writeSurfacePixel (surface< void, cudaSurfaceType2DLayered > surf, int x, int y, int z, T value) [inline]

```
244     {
245         surf2DLayeredwrite(value, surf, sizeof(T)*x, y, z, cudaBoundaryModeTrap);
246     }
```

10.32.2 Member Data Documentation

10.32.2.1 template<typename T> int Image4DCudaArray<T>::KernelInst::imgh

10.32.2.2 template<typename T> int Image4DCudaArray<T>::KernelInst::imgw

10.32.2.3 template<typename T> int Image4DCudaArray<T>::KernelInst::layerw

The documentation for this struct was generated from the following file:

- cudatrack/cudalImageList.h

10.33 Image4DMemory< T >::KernelParams Struct Reference

```
#include <cudaImageList.h>
```

Public Member Functions

- CUBOTH int2 GetImagePos (int z, int l)

Public Attributes

- T * d_data
- size_t pitch
- int cols
- int depth
- int imgw
- int imgh

10.33.1 Member Function Documentation

10.33.1.1 template<typename T > CUBOTH int2 Image4DMemory< T >::KernelParams::GetImagePos (int z, int l)
[inline]

```
414             {
415         int img = z+depth*l;
416         return make_int2( (img % cols) * imgw, (img / cols) * imgh);
417     }
```

10.33.2 Member Data Documentation

10.33.2.1 template<typename T > int Image4DMemory< T >::KernelParams::cols

10.33.2.2 template<typename T > T* Image4DMemory< T >::KernelParams::d_data

10.33.2.3 template<typename T > int Image4DMemory< T >::KernelParams::depth

10.33.2.4 template<typename T > int Image4DMemory< T >::KernelParams::imgh

10.33.2.5 template<typename T > int Image4DMemory< T >::KernelParams::imgw

10.33.2.6 template<typename T > size_t Image4DMemory< T >::KernelParams::pitch

The documentation for this struct was generated from the following file:

- cudatrack/cudalImageList.h

10.34 QueuedCUDATracker::KernelProfileTime Struct Reference

```
#include <QueuedCUDATracker.h>
```

Public Member Functions

- KernelProfileTime ()

Public Attributes

- double com
- double qi
- double imageCopy
- double zcompute
- double zlutAlign
- double getResults

10.34.1 Constructor & Destructor Documentation

10.34.1.1 QueuedCUDATracker::KernelProfileTime::KernelProfileTime () [inline]

```
225 {com=qi=imageCopy=zcompute=zlutAlign=getResults=0.0;}
```

10.34.2 Member Data Documentation

10.34.2.1 double QueuedCUDATracker::KernelProfileTime::com

10.34.2.2 double QueuedCUDATracker::KernelProfileTime::getResults

10.34.2.3 double QueuedCUDATracker::KernelProfileTime::imageCopy

10.34.2.4 double QueuedCUDATracker::KernelProfileTime::qi

10.34.2.5 double QueuedCUDATracker::KernelProfileTime::zcompute

10.34.2.6 double QueuedCUDATracker::KernelProfileTime::zlutAlign

The documentation for this struct was generated from the following file:

- cudatrac/cuadatrac/QueuedCUDATracker.h

10.35 kissfft< T_Scalar, T_traits > Class Template Reference

```
#include <kissfft.h>
```

Public Types

- **typedef T_traits traits_type**
- **typedef traits_type::scalar_type scalar_type**
- **typedef traits_type::cpx_type cpx_type**

Public Member Functions

- **kissfft** (int **nfft**, bool **inverse**, const **traits_type** &**traits**=**traits_type()**)
- **void transform** (const **cpx_type** ***src**, **cpx_type** ***dst**)
- **int nfft ()**

Private Member Functions

- void `kf_work` (int stage, `cpx_type` *Fout, const `cpx_type` *f, size_t fstride, size_t in_stride)
- void `C_ADD` (`cpx_type` &c, const `cpx_type` &a, const `cpx_type` &b)
- void `C_MUL` (`cpx_type` &c, const `cpx_type` &a, const `cpx_type` &b)
- void `C_SUB` (`cpx_type` &c, const `cpx_type` &a, const `cpx_type` &b)
- void `C_ADDTO` (`cpx_type` &c, const `cpx_type` &a)
- void `C_FIXDIV` (`cpx_type` &, int)
- `scalar_type S_MUL` (const `scalar_type` &a, const `scalar_type` &b)
- `scalar_type HALF_OF` (const `scalar_type` &a)
- void `C_MULBYSCALAR` (`cpx_type` &c, const `scalar_type` &a)
- void `kf_bfly2` (`cpx_type` *Fout, const size_t fstride, int m)
- void `kf_bfly4` (`cpx_type` *Fout, const size_t fstride, const size_t m)
- void `kf_bfly3` (`cpx_type` *Fout, const size_t fstride, const size_t m)
- void `kf_bfly5` (`cpx_type` *Fout, const size_t fstride, const size_t m)
- void `kf_bfly_generic` (`cpx_type` *Fout, const size_t fstride, int m, int p)

Private Attributes

- int `_nfft`
- bool `_inverse`
- std::vector< `cpx_type` > `_twiddles`
- std::vector< int > `_stageRadix`
- std::vector< int > `_stageRemainder`

10.35.1 Member Typedef Documentation

10.35.1.1 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> typedef traits_type::cpx_type `kissfft< T_Scalar, T_traits >::cpx_type`

10.35.1.2 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> typedef traits_type::scalar_type `kissfft< T_Scalar, T_traits >::scalar_type`

10.35.1.3 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> typedef T_traits `kissfft< T_Scalar, T_traits >::traits_type`

10.35.2 Constructor & Destructor Documentation

10.35.2.1 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> `kissfft< T_Scalar, T_traits >::kissfft` (int `nfft`, bool `inverse`, const traits_type & `traits` = traits_type ()) [inline]

```
65     : _nfft(nfft), _inverse(inverse)
66     {
67         traits.prepare(_twiddles, _nfft, _inverse,
68         _stageRadix, _stageRemainder);
69     }
```

10.35.3 Member Function Documentation

10.35.3.1 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> void `kissfft< T_Scalar, T_traits >::C_ADD` (`cpx_type` & `c`, const `cpx_type` & `a`, const `cpx_type` & `b`) [inline], [private]

```
114 { c=a+b; }
```

10.35.3.2 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> void kissfft< T_Scalar, T_traits >::C_ADDTO (cpx_type & c, const cpx_type & a) [inline], [private]

```
117 { c+=a; }
```

10.35.3.3 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> void kissfft< T_Scalar, T_traits >::C_FIXDIV (cpx_type &, int) [inline], [private]

```
118 {} // NO-OP for float types
```

10.35.3.4 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> void kissfft< T_Scalar, T_traits >::C_MUL (cpx_type & c, const cpx_type & a, const cpx_type & b) [inline], [private]

```
115 { c=a*b; }
```

10.35.3.5 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> void kissfft< T_Scalar, T_traits >::C_MULBYSCALAR (cpx_type & c, const scalar_type & a) [inline], [private]

```
121 { c*=a; }
```

10.35.3.6 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> void kissfft< T_Scalar, T_traits >::C_SUB (cpx_type & c, const cpx_type & a, const cpx_type & b) [inline], [private]

```
116 { c=a-b; }
```

10.35.3.7 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> scalar_type kissfft< T_Scalar, T_traits >::HALF_OF (const scalar_type & a) [inline], [private]

```
120 { return a*.5; }
```

10.35.3.8 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> void kissfft< T_Scalar, T_traits >::kf_bfly2 (cpx_type * Fout, const size_t fstride, int m) [inline], [private]

```
124 {
125     for (int k=0;k<m;++k) {
126         cpx_type t = Fout[m+k] * _twiddles[k*fstride];
127         Fout[m+k] = Fout[k] - t;
128         Fout[k] += t;
129     }
130 }
```

10.35.3.9 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> void kissfft< T_Scalar, T_traits >::kf_bfly3(cpx_type * Fout, const size_t fstride, const size_t m) [inline], [private]

```

155     {
156         size_t k=m;
157         const size_t m2 = 2*m;
158         cpx_type *tw1,*tw2;
159         cpx_type scratch[5];
160         cpx_type epi3;
161         epi3 = _twiddles[fstride*m];
162
163         tw1=tw2=&_twiddles[0];
164
165         do{
166             C_FIXDIV(*Fout,3); C_FIXDIV(Fout[m],3); C_FIXDIV(Fout[m2],3);
167
168             C_MUL(scratch[1],Fout[m] , *tw1);
169             C_MUL(scratch[2],Fout[m2] , *tw2);
170
171             C_ADD(scratch[3],scratch[1],scratch[2]);
172             C_SUB(scratch[0],scratch[1],scratch[2]);
173             tw1 += fstride;
174             tw2 += fstride*2;
175
176             Fout[m] = cpx_type( Fout->real() - HALF_OF(scratch[3].real()) , Fout->imag(
177             ) - HALF_OF(scratch[3].imag()) );
178
179             C_MULBYSCALAR( scratch[0] , epi3.imag() );
180
181             C_ADDTO(*Fout,scratch[3]);
182
183             Fout[m2] = cpx_type( Fout[m].real() + scratch[0].imag() , Fout[m].imag() - scratch
184             [0].real() );
185
186             C_ADDTO( Fout[m] , cpx_type( -scratch[0].imag(),scratch[0].real() ) );
187             ++Fout;
188         }while(--k);
189     }

```

10.35.3.10 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> void kissfft< T_Scalar, T_traits >::kf_bfly4(cpx_type * Fout, const size_t fstride, const size_t m) [inline], [private]

```

133     {
134         cpx_type scratch[7];
135         int negative_if_inverse = _inverse * -2 +1;
136         for (size_t k=0;k<m;++k) {
137             scratch[0] = Fout[k+m] * _twiddles[k*fstride];
138             scratch[1] = Fout[k+2*m] * _twiddles[k*fstride*2];
139             scratch[2] = Fout[k+3*m] * _twiddles[k*fstride*3];
140             scratch[5] = Fout[k] - scratch[1];
141
142             Fout[k] += scratch[1];
143             scratch[3] = scratch[0] + scratch[2];
144             scratch[4] = scratch[0] - scratch[2];
145             scratch[4] = cpx_type( scratch[4].imag()*negative_if_inverse , -scratch[4].real()*
negative_if_inverse );
146
147             Fout[k+2*m] = Fout[k] - scratch[3];
148             Fout[k] += scratch[3];
149             Fout[k+m] = scratch[5] + scratch[4];
150             Fout[k+3*m] = scratch[5] - scratch[4];
151         }
152     }

```

10.35.3.11 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> void kissfft< T_Scalar, T_traits >::kf_bfly5(cpx_type * Fout, const size_t fstride, const size_t m) [inline], [private]

```

190     {
191         cpx_type *Fout0,*Fout1,*Fout2,*Fout3,*Fout4;
192         size_t u;
193         cpx_type scratch[13];
194         cpx_type * twiddles = &_twiddles[0];
195         cpx_type *tw;
196         cpx_type ya,yb;

```

```

197     ya = twiddles[fstride*m];
198     yb = twiddles[fstride*2*m];
199
200     Fout0=Fout;
201     Fout1=Fout0+m;
202     Fout2=Fout0+2*m;
203     Fout3=Fout0+3*m;
204     Fout4=Fout0+4*m;
205
206     tw=twiddles;
207     for ( u=0; u<m; ++u ) {
208         C_FIXDIV( *Fout0,5); C_FIXDIV( *Fout1,5);
209         C_FIXDIV( *Fout2,5); C_FIXDIV( *Fout3,5); C_FIXDIV( *Fout4,5);
210         scratch[0] = *Fout0;
211
212         C_MUL(scratch[1], *Fout1, tw[u*fstride]);
213         C_MUL(scratch[2], *Fout2, tw[2*u*fstride]);
214         C_MUL(scratch[3], *Fout3, tw[3*u*fstride]);
215         C_MUL(scratch[4], *Fout4, tw[4*u*fstride]);
216
217         C_ADD( scratch[7], scratch[1], scratch[4]);
218         C_SUB( scratch[10], scratch[1], scratch[4]);
219         C_ADD( scratch[8], scratch[2], scratch[3]);
220         C_SUB( scratch[9], scratch[2], scratch[3]);
221
222         C_ADDTO( *Fout0, scratch[7]);
223         C_ADDTO( *Fout0, scratch[8]);
224
225         scratch[5] = scratch[0] + cpx_type(
226             S_MUL(scratch[7].real(), ya.real()) + S_MUL(scratch[8].real(), yb.real())
227         ),
228             S_MUL(scratch[7].imag(), ya.real()) + S_MUL(scratch[8].imag(), yb.real())
229         );
230
231         scratch[6] = cpx_type(
232             S_MUL(scratch[10].imag(), ya.imag()) + S_MUL(scratch[9].imag(), yb.imag()),
233             -S_MUL(scratch[10].real(), ya.imag()) - S_MUL(scratch[9].real(), yb.imag())
234         );
235
236         C_SUB(*Fout1,scratch[5],scratch[6]);
237         C_ADD(*Fout4,scratch[5],scratch[6]);
238
239         scratch[11] = scratch[0] +
240             cpx_type(
241                 S_MUL(scratch[7].real(), yb.real()) + S_MUL(scratch[8].real(), ya.real()
242             ),
243                 S_MUL(scratch[7].imag(), yb.real()) + S_MUL(scratch[8].imag(), ya.real()
244             );
245
246         scratch[12] = cpx_type(
247             -S_MUL(scratch[10].imag(), yb.imag()) + S_MUL(scratch[9].imag(), ya.imag())
248             ,
249                 S_MUL(scratch[10].real(), yb.imag()) - S_MUL(scratch[9].real(), ya.imag()
250             );
251
252         }
253     }

```

10.35.3.12 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> void kissfft< T_Scalar, T_traits >::kf_bfly_generic (cpx_type * Fout, const size_t fstride, int m, int p) [inline], [private]

```

262     {
263         int u,k,q1,q;
264         cpx_type * twiddles = &_twiddles[0];
265         cpx_type t;
266         int Norig = _nfft;
267         cpx_type *scratchbuf = (cpx_type*) ALLOCA(sizeof(
268             cpx_type)*p);
269         for ( u=0; u<m; ++u ) {
270             k=u;
271             for ( q1=0 ; q1<p ; ++q1 ) {
272                 scratchbuf[q1] = Fout[ k ];
273                 C_FIXDIV(scratchbuf[q1],p);
274                 k += m;

```

```

275
276
277     k=u;
278     for ( q1=0 ; q1<p ; ++q1 ) {
279         int twidx=0;
280         Fout[ k ] = scratchbuf[0];
281         for (q=1;q<p;++q) {
282             twidx += fstride * k;
283             if (twidx>=Norig) twidx-=Norig;
284             C_MUL(t,scratchbuf[q] , twiddles[twidx] );
285             C_ADDTO( Fout[ k ] ,t);
286         }
287         k += m;
288     }
289 }
290 }
```

10.35.3.13 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> void kissfft< T_Scalar, T_traits >::kf_work(int stage, cpx_type * Fout, const cpx_type * f, size_t fstride, size_t in_stride) [inline], [private]

```

79     {
80         int p = _stageRadix[stage];
81         int m = _stageRemainder[stage];
82         cpx_type * Fout_beg = Fout;
83         cpx_type * Fout_end = Fout + p*m;
84
85         if (m==1) {
86             do{
87                 *Fout = *f;
88                 f += fstride*in_stride;
89             }while(++Fout != Fout_end );
90         }else{
91             do{
92                 // recursive call:
93                 // DFT of size m*p performed by doing
94                 // p instances of smaller DFTs of size m,
95                 // each one takes a decimated version of the input
96                 kf_work(stage+1, Fout , f, fstride*p,in_stride);
97                 f += fstride*in_stride;
98             }while( (Fout += m) != Fout_end );
99         }
100
101         Fout=Fout_beg;
102
103         // recombine the p smaller DFTs
104         switch (p) {
105             case 2: kf_bfly2(Fout,fstride,m); break;
106             case 3: kf_bfly3(Fout,fstride,m); break;
107             case 4: kf_bfly4(Fout,fstride,m); break;
108             case 5: kf_bfly5(Fout,fstride,m); break;
109             default: kf_bfly_generic(Fout,fstride,m,p); break;
110         }
111     }
```

10.35.3.14 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> int kissfft< T_Scalar, T_traits >::nfft() [inline]

```
75 { return _nfft; }
```

10.35.3.15 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> scalar_type kissfft< T_Scalar, T_traits >::S_MUL(const scalar_type & a, const scalar_type & b) [inline], [private]

```
119 { return a*b; }
```

10.35.3.16 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> void kissfft< T_Scalar, T_traits >::transform (const cpx_type * src, cpx_type * dst) [inline]

```
71     {
72         kf_work(0, dst, src, 1,1);
73     }
```

10.35.4 Member Data Documentation

10.35.4.1 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> bool kissfft< T_Scalar, T_traits >::_inverse [private]

10.35.4.2 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> int kissfft< T_Scalar, T_traits >::_nfft [private]

10.35.4.3 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> std::vector<int> kissfft< T_Scalar, T_traits >::_stageRadix [private]

10.35.4.4 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> std::vector<int> kissfft< T_Scalar, T_traits >::_stageRemainder [private]

10.35.4.5 template<typename T_Scalar, typename T_traits = kissfft_utils::traits<T_Scalar>> std::vector<cpx_type> kissfft< T_Scalar, T_traits >::_twiddles [private]

The documentation for this class was generated from the following file:

- cputrack/kissfft.h

10.36 LocalizationJob Struct Reference

Structure for region of interest metadata.

```
#include <qtrk_c_api.h>
```

Public Member Functions

- [LocalizationJob \(\)](#)
- [LocalizationJob \(uint frame, uint timestamp, uint zlutPlane, uint zlutIndex\)](#)

Public Attributes

- **uint frame**
Frame number this ROI belongs to.
- **uint timestamp**
Time stamp of the frame.
- **int zlutIndex**
Read number of this ROI. Used to get the right ZLUT from memory.
- **vector3f initialPos**
Optional (Not used)

10.36.1 Detailed Description

Structure for region of interest metadata.

Compiled without padding to line up with LabVIEW alignment. Size is 24 bytes.

Warning

Changing this requires changing of the linked LabVIEW cluster QTrkLocalizationJob.ctl.

Note

Frame and timestamp are ignored by tracking code itself, but usable for the calling code.

10.36.2 Constructor & Destructor Documentation

10.36.2.1 LocalizationJob::LocalizationJob() [inline]

```
42         {
43             frame=timestamp=zlutIndex=0;
44         }
```

10.36.2.2 LocalizationJob::LocalizationJob(uint frame, uint timestamp, uint zlutPlane, uint zlutIndex) [inline]

```
45         :
46         frame (frame), timestamp(timestamp),
47         zlutIndex(zlutIndex)
48     {}
```

10.36.3 Member Data Documentation

10.36.3.1 uint LocalizationJob::frame

Frame number this ROI belongs to.

10.36.3.2 vector3f LocalizationJob::initialPos

Optional (Not used)

10.36.3.3 uint LocalizationJob::timestamp

Time stamp of the frame.

10.36.3.4 int LocalizationJob::zlutIndex

Bead number of this ROI. Used to get the right ZLUT from memory.

The documentation for this struct was generated from the following file:

- cputrack/qtrk_c_api.h

10.37 LocalizationParams Struct Reference

```
#include <QueuedCUDATracker.h>
```

Public Attributes

- int [zlutIndex](#)
- int [zlutPlane](#)

10.37.1 Member Data Documentation

10.37.1.1 int LocalizationParams::zlutIndex

10.37.1.2 int LocalizationParams::zlutPlane

The documentation for this struct was generated from the following file:

- cudatrack/[QueuedCUDATracker.h](#)

10.38 LocalizationResult Struct Reference

Structure for job results.

```
#include <qtrk_c_api.h>
```

Public Member Functions

- [vector2f pos2D \(\)](#)

Final 2D position found.

Public Attributes

- **LocalizationJob job**
Job metadata. See [LocalizationJob](#).
- **vector3f pos**
Final 3D position found. If no z localization was performed, the value of z will be 0.
- **vector2f firstGuess**
(x,y) position found by the COM localization. Used as initial position for the subsequent algorithms.
- **uint error**
Flag (boolean) indicating whether the ROI boundary was hit during localization. A 1 indicates a hit.
- **float imageMean**
Average pixel value of the ROI.

10.38.1 Detailed Description

Structure for job results.

Compiled without padding to line up with LabVIEW alignment. Size is 52 bytes.

Warning

Changing this requires changing of the linked LabVIEW cluster QTrkLocalizationResult.ctl.

10.38.2 Member Function Documentation

10.38.2.1 vector2f LocalizationResult::pos2D() [inline]

Final 2D position found.

10.38.3 Member Data Documentation

10.38.3.1 uint LocalizationResult::error

Flag (boolean) indicating whether the ROI boundary was hit during localization. A 1 indicates a hit.

10.38.3.2 vector2f LocalizationResult::firstGuess

(x,y) position found by the COM localization. Used as initial position for the subsequent algorithms.

10.38.3.3 float LocalizationResult::imageMean

Average pixel value of the ROI.

10.38.3.4 LocalizationJob LocalizationResult::job

Job metadata. See [LocalizationJob](#).

10.38.3.5 vector3f LocalizationResult::pos

Final 3D position found. If no z localization was performed, the value of z will be 0.

The documentation for this struct was generated from the following file:

- cputrack/qtrk_c_api.h

10.39 LsqSqQuadFit< T > Class Template Reference

```
#include <LsqQuadraticFit.h>
```

Classes

- struct [Coeff](#)

Public Member Functions

- [CUDA_SUPPORTED_FUNC LsqSqQuadFit \(uint numPts, const T *xval, const T *yval, const T *weights=0\)](#)
- [CUDA_SUPPORTED_FUNC LsqSqQuadFit \(\)](#)
- [CUDA_SUPPORTED_FUNC void calculate \(uint numPts, const T *X, const T *Y, const T *weights\)](#)
- [CUDA_SUPPORTED_FUNC T compute \(T pos\)](#)
- [CUDA_SUPPORTED_FUNC T computeDeriv \(T pos\)](#)
- [CUDA_SUPPORTED_FUNC T maxPos \(\)](#)
- [CUDA_SUPPORTED_FUNC T vertexForm \(\)](#)

Public Attributes

- T [a](#)
- T [b](#)
- T [c](#)
- T [d](#)
- T [h](#)
- T [k](#)
- float [xoffset](#)

Private Member Functions

- [CUDA_SUPPORTED_FUNC Coeff computeSums \(const T *X, const T *Y, const T *weights, uint numPts\)](#)

10.39.1 Constructor & Destructor Documentation

10.39.1.1 template<typename T> CUDA_SUPPORTED_FUNC LsqSqQuadFit< T >::LsqSqQuadFit (uint numPts, const T * xval, const T * yval, const T * weights = 0) [inline]

```
28      {
29          calculate(numPts, xval, yval, weights);
30          xoffset =0;
31      }
```

10.39.1.2 template<typename T> CUDA_SUPPORTED_FUNC LsqSqQuadFit< T >::LsqSqQuadFit()
[inline]

```
34     {
35         a=b=c=d=0;
36         xoffset =0;
37     }
```

10.39.2 Member Function Documentation

10.39.2.1 template<typename T> CUDA_SUPPORTED_FUNC void LsqSqQuadFit< T >::calculate(uint numPts,
const T * X, const T * Y, const T * weights) [inline]

```
40     {
41         Coeff co = computeSums(X, Y, weights, numPts);
42         co.abc(a,b,c,d);
43     }
```

10.39.2.2 template<typename T> CUDA_SUPPORTED_FUNC T LsqSqQuadFit< T >::compute(T pos)
[inline]

```
46     {
47         pos -= xoffset;
48         return a*pos*pos + b*pos + c;
49     }
```

10.39.2.3 template<typename T> CUDA_SUPPORTED_FUNC T LsqSqQuadFit< T >::computeDeriv(T pos)
[inline]

```
52     {
53         pos -= xoffset;
54         return 2*a*pos + b;
55     }
```

10.39.2.4 template<typename T> CUDA_SUPPORTED_FUNC Coeff LsqSqQuadFit< T >::computeSums(const T
* X, const T * Y, const T * weights, uint numPts) [inline], [private]

```
73     {
74         //notation sjk to mean the sum of x_i^j*y_i^k.
75         /* s40 = getSx4(); //sum of x^4
76         s30 = getSx3(); //sum of x^3
77         s20 = getSx2(); //sum of x^2
78         s10 = getSx(); //sum of x
79
80         s21 = getSx2y(); //sum of x^2*y
81         s11 = getSxy(); //sum of x*y
82         s01 = getSy(); //sum of y
83         */
84
85         if (weights) {
86             T Sx = 0, Sy = 0;
87             T Sx2 = 0, Sx3 = 0;
88             T Sxy = 0, Sx4=0, Sx2y=0;
89             T Sw = 0;
90             for (uint i=0;i<numPts;i++)
91             {
92                 T x = X[i];
93                 T y = Y[i];
94                 Sx += x*weights[i];
95                 Sy += y*weights[i];
96             }
97         }
98     }
```

```

97         T sq = x*x;
98         Sx2 += x*x*weights[i];
99         Sx3 += sq*x*weights[i];
100        Sx4 += sq*sq*weights[i];
101        Sxy += x*y*weights[i];
102        Sx2y += sq*y*weights[i];
103        Sw += weights[i];
104    }
105
106    Coeff co;
107    co.s10 = Sx; co.s20 = Sx2; co.s30 = Sx3; co.s40 = Sx4;
108    co.s01 = Sy; co.s11 = Sxy; co.s21 = Sx2y;
109    co.s00 = Sw;
110    return co;
111 } else {
112     T Sx = 0, Sy = 0;
113     T Sx2 = 0, Sx3 = 0;
114     T Sxy = 0, Sx4=0, Sx2y=0;
115     for (uint i=0;i<numPts;i++)
116     {
117         T x = X[i];
118         T y = Y[i];
119         Sx += x;
120         Sy += y;
121         T sq = x*x;
122         Sx2 += x*x;
123         Sx3 += sq*x;
124         Sx4 += sq*sq;
125         Sxy += x*y;
126         Sx2y += sq*y;
127     }
128
129     Coeff co;
130     co.s10 = Sx; co.s20 = Sx2; co.s30 = Sx3; co.s40 = Sx4;
131     co.s01 = Sy; co.s11 = Sxy; co.s21 = Sx2y;
132     co.s00 = numPts;
133     return co;
134 }
135 }
```

10.39.2.5 template<typename T> CUDA_SUPPORTED_FUNC T LsqSqQuadFit< T >::maxPos() [inline]

```

58     {
59         return -b/(2*a);
60     }
```

10.39.2.6 template<typename T> CUDA_SUPPORTED_FUNC T LsqSqQuadFit< T >::vertexForm() [inline]

```

63     {
64         // Find equation in form a(x-h)^2+k
65         h = -b/(2*a);
66         k = c - a*h*h;
67         return k;
68     }
```

10.39.3 Member Data Documentation

10.39.3.1 template<typename T> T LsqSqQuadFit< T >::a

10.39.3.2 template<typename T> T LsqSqQuadFit< T >::b

10.39.3.3 template<typename T> T LsqSqQuadFit< T >::c

10.39.3.4 template<typename T> T LsqSqQuadFit< T >::d

10.39.3.5 template<typename T> T LsqSqQuadFit< T >::h

10.39.3.6 template<typename T> T LsqSqQuadFit< T >::k

10.39.3.7 template<typename T> float LsqSqQuadFit< T >::xoffset

The documentation for this class was generated from the following file:

- cputrack/LsqQuadraticFit.h

10.40 LVArray< T > Struct Template Reference

```
#include <labview.h>
```

Public Attributes

- int32_t dimSize
- T elem [1]

10.40.1 Member Data Documentation

10.40.1.1 template<typename T> int32_t LVArray< T >::dimSize

10.40.1.2 template<typename T> T LVArray< T >::elem[1]

The documentation for this struct was generated from the following file:

- cputrack/labview.h

10.41 LVArray2D< T > Struct Template Reference

```
#include <labview.h>
```

Public Member Functions

- T & xy (int col, int row)
- T & get (int row, int col)
- int numElem ()

Public Attributes

- int32_t dimSizes [2]
- T elem [1]

10.41.1 Member Function Documentation

10.41.1.1 template<typename T> T& LVArray2D< T >::get(int row, int col) [inline]

```
44 {
45     return elem[row*dimSizes[1]+col];
46 }
```

10.41.1.2 template<typename T> int LVArray2D< T >::numElem() [inline]

```
47 { return dimSizes[0]*dimSizes[1]; }
```

10.41.1.3 template<typename T> T& LVArray2D< T >::xy(int col, int row) [inline]

```
41 {
42     return elem[row*dimSizes[1]+col];
43 }
```

10.41.2 Member Data Documentation

10.41.2.1 template<typename T> int32_t LVArray2D< T >::dimSizes[2]

10.41.2.2 template<typename T> T LVArray2D< T >::elem[1]

The documentation for this struct was generated from the following file:

- cputrack/labview.h

10.42 LVArray3D< T > Struct Template Reference

```
#include <labview.h>
```

Public Member Functions

- int numElem ()

Public Attributes

- int32_t dimSizes [3]
- T elem [1]

10.42.1 Member Function Documentation

10.42.1.1 template<typename T> int LVArray3D< T >::numElem() [inline]

```
55 { return dimSizes[0]*dimSizes[1]*dimSizes[2]; }
```

10.42.2 Member Data Documentation

10.42.2.1 template<typename T> int32_t LVArray3D< T >::dimSizes[3]

10.42.2.2 template<typename T> T LVArray3D< T >::elem[1]

The documentation for this struct was generated from the following file:

- cputrack/labview.h

10.43 LVArrayND< T, N > Struct Template Reference

```
#include <labview.h>
```

Public Member Functions

- int numElem()

Public Attributes

- int32_t dimSizes[N]
- T elem[1]

10.43.1 Member Function Documentation

10.43.1.1 template<typename T, int N> int LVArrayND< T, N >::numElem() [inline]

```
63 {
64     int n = dimSizes[0];
65     for (int i=1;i<N;i++) n*=dimSizes[i];
66     return n;
67 }
```

10.43.2 Member Data Documentation

10.43.2.1 template<typename T, int N> int32_t LVArrayND< T, N >::dimSizes[N]

10.43.2.2 template<typename T, int N> T LVArrayND< T, N >::elem[1]

The documentation for this struct was generated from the following file:

- cputrack/labview.h

10.44 LVDataType< T > Struct Template Reference

```
#include <labview.h>
```

The documentation for this struct was generated from the following file:

- cputrack/labview.h

10.45 LVDataType< double > Struct Template Reference

```
#include <labview.h>
```

Public Types

- enum { **code** =10 }

10.45.1 Member Enumeration Documentation

10.45.1.1 anonymous enum

Enumerator

code

```
75 { enum { code=10 }; };
```

The documentation for this struct was generated from the following file:

- cputrack/labview.h

10.46 LVDataType< float > Struct Template Reference

```
#include <labview.h>
```

Public Types

- enum { **code** =9 }

10.46.1 Member Enumeration Documentation

10.46.1.1 anonymous enum

Enumerator

code

```
74 { enum { code=9 }; };
```

The documentation for this struct was generated from the following file:

- cputrack/labview.h

10.47 LVDataType< int16_t > Struct Template Reference

```
#include <labview.h>
```

Public Types

- enum { **code** =2 }

10.47.1 Member Enumeration Documentation

10.47.1.1 anonymous enum

Enumerator

code

```
77 { enum { code=2 }; };
```

The documentation for this struct was generated from the following file:

- cputrack/labview.h

10.48 LVDataType< int32_t > Struct Template Reference

```
#include <labview.h>
```

Public Types

- enum { **code** =3 }

10.48.1 Member Enumeration Documentation

10.48.1.1 anonymous enum

Enumerator

code

```
78 { enum { code=3 }; };
```

The documentation for this struct was generated from the following file:

- cputrack/labview.h

10.49 LVDataType< int64_t > Struct Template Reference

```
#include <labview.h>
```

Public Types

- enum { code =4 }

10.49.1 Member Enumeration Documentation

10.49.1.1 anonymous enum

Enumerator

code

```
79 { enum { code=4 }; };
```

The documentation for this struct was generated from the following file:

- cputrack/labview.h

10.50 LVDataType< int8_t > Struct Template Reference

```
#include <labview.h>
```

Public Types

- enum { code =1 }

10.50.1 Member Enumeration Documentation

10.50.1.1 anonymous enum

Enumerator

code

```
76 { enum { code=1 }; };
```

The documentation for this struct was generated from the following file:

- cputrack/labview.h

10.51 LVDataType< std::complex< double > > Struct Template Reference

```
#include <labview.h>
```

Public Types

- enum { **code** =0xd }

10.51.1 Member Enumeration Documentation

10.51.1.1 anonymous enum

Enumerator

code

```
85 { enum { code=0xd }; };
```

The documentation for this struct was generated from the following file:

- cputrack/labview.h

10.52 LVDataType< std::complex< float > > Struct Template Reference

```
#include <labview.h>
```

Public Types

- enum { **code** =0xc }

10.52.1 Member Enumeration Documentation

10.52.1.1 anonymous enum

Enumerator

code

```
84 { enum { code=0xc }; };
```

The documentation for this struct was generated from the following file:

- cputrack/labview.h

10.53 LVDataType< uint16_t > Struct Template Reference

```
#include <labview.h>
```

Public Types

- enum { **code** =6 }

10.53.1 Member Enumeration Documentation

10.53.1.1 anonymous enum

Enumerator

code

```
81 { enum { code=6 }; };
```

The documentation for this struct was generated from the following file:

- cputrack/labview.h

10.54 LVDataType< uint32_t > Struct Template Reference

```
#include <labview.h>
```

Public Types

- enum { **code** =7 }

10.54.1 Member Enumeration Documentation

10.54.1.1 anonymous enum

Enumerator

code

```
82 { enum { code=7 }; };
```

The documentation for this struct was generated from the following file:

- cputrack/labview.h

10.55 LVDataType< uint64_t > Struct Template Reference

```
#include <labview.h>
```

Public Types

- enum { **code** =8 }

10.55.1 Member Enumeration Documentation

10.55.1.1 anonymous enum

Enumerator

code

```
83 { enum { code=8 }; };
```

The documentation for this struct was generated from the following file:

- cputrack/labview.h

10.56 LVDataType< uint8_t > Struct Template Reference

```
#include <labview.h>
```

Public Types

- enum { **code** =5 }

10.56.1 Member Enumeration Documentation

10.56.1.1 anonymous enum

Enumerator

code

```
80 { enum { code=5 }; };
```

The documentation for this struct was generated from the following file:

- cputrack/labview.h

10.57 Matrix3X3 Class Reference

```
#include <utils.h>
```

Public Member Functions

- [Matrix3X3 \(\)](#)
- [Matrix3X3 \(vector3f x, vector3f y, vector3f z\)](#)
- [vector3f diag \(\) const](#)
- [float & operator\[\] \(int i\)](#)
- [const float & operator\[\] \(int i\) const](#)
- [vector3f & row \(int i\)](#)
- [const vector3f & row \(int i\) const](#)
- [float & operator\(\) \(int i, int j\)](#)
- [const float & operator\(\) \(int i, int j\) const](#)
- [float & at \(int i, int j\)](#)
- [const float & at \(int i, int j\) const](#)
- [float Determinant \(\) const](#)
- [Matrix3X3 Inverse \(\) const](#)
- [Matrix3X3 InverseTranspose \(\) const](#)
- [Matrix3X3 & operator*=\(float a\)](#)
- [Matrix3X3 & operator+=\(const Matrix3X3 &b\)](#)
- [void dbgprint \(\)](#)

Static Public Member Functions

- [static void test \(\)](#)

Public Attributes

- [float m \[9\]](#)

10.57.1 Constructor & Destructor Documentation

10.57.1.1 Matrix3X3::Matrix3X3() [inline]

```
270 { for(int i=0;i<9;i++) m[i]=0.0f; }
```

10.57.1.2 Matrix3X3::Matrix3X3(vector3f x, vector3f y, vector3f z) [inline]

```
271 {  
272     row(0) = x;  
273     row(1) = y;  
274     row(2) = z;  
275 }
```

10.57.2 Member Function Documentation

10.57.2.1 float& Matrix3X3::at(int i, int j) [inline]

```
288 { return m[3*i+j]; }
```

10.57.2.2 const float& Matrix3X3::at(int i, int j) const [inline]

```
289 { return m[3*i+j]; }
```

10.57.2.3 void Matrix3X3::dbgprint() [inline]

```
361 {  
362     dbgprintf("%.f\t%.f\t%.f\n", m[0],m[1],m[2]);  
363     dbgprintf("%.f\t%.f\t%.f\n", m[3],m[4],m[5]);  
364     dbgprintf("%.f\t%.f\t%.f\n", m[6],m[7],m[8]);  
365 }
```

10.57.2.4 float Matrix3X3::Determinant() const [inline]

```
292 {  
293     return at(0,0)*(at(1,1)*at(2,2)-at(2,1)*at(1,2))  
294         -at(0,1)*(at(1,0)*at(2,2)-at(1,2)*at(2,0))  
295         +at(0,2)*(at(1,0)*at(2,1)-at(1,1)*at(2,0));  
296 }
```

10.57.2.5 vector3f Matrix3X3::diag() const [inline]

```
277 { return vector3f(at(0,0),at(1,1),at(2,2)); }
```

10.57.2.6 Matrix3X3 Matrix3X3::Inverse() const [inline]

```

299     {
300         float det = Determinant();
301         if (det != 0.0f) {
302             float invdet = 1/det;
303             Matrix3X3 result;
304             result(0,0) = (at(1,1)*at(2,2)-at(2,1)*at(1,2))*invdet;
305             result(1,0) = -(at(0,1)*at(2,2)-at(0,2)*at(2,1))*invdet;
306             result(2,0) = (at(0,1)*at(1,2)-at(0,2)*at(1,1))*invdet;
307             result(0,1) = -(at(1,0)*at(2,2)-at(1,2)*at(2,0))*invdet;
308             result(1,1) = (at(0,0)*at(2,2)-at(0,2)*at(2,0))*invdet;
309             result(2,1) = -(at(0,0)*at(1,2)-at(1,0)*at(0,2))*invdet;
310             result(0,2) = (at(1,0)*at(2,1)-at(2,0)*at(1,1))*invdet;
311             result(1,2) = -(at(0,0)*at(2,1)-at(2,0)*at(0,1))*invdet;
312             result(2,2) = (at(0,0)*at(1,1)-at(1,0)*at(0,1))*invdet;
313             return result;
314     }
315     return Matrix3X3();
316 }
```

10.57.2.7 Matrix3X3 Matrix3X3::InverseTranspose() const [inline]

```

318     {
319         float det = Determinant();
320         if (det != 0.0f) {
321             float invdet = 1/det;
322             Matrix3X3 result;
323             result(0,0) = (at(1,1)*at(2,2)-at(2,1)*at(1,2))*invdet;
324             result(0,1) = -(at(0,1)*at(2,2)-at(0,2)*at(2,1))*invdet;
325             result(0,2) = (at(0,1)*at(1,2)-at(0,2)*at(1,1))*invdet;
326             result(1,0) = -(at(1,0)*at(2,2)-at(1,2)*at(2,0))*invdet;
327             result(1,1) = (at(0,0)*at(2,2)-at(0,2)*at(2,0))*invdet;
328             result(1,2) = -(at(0,0)*at(1,2)-at(1,0)*at(0,2))*invdet;
329             result(2,0) = (at(1,0)*at(2,1)-at(2,0)*at(1,1))*invdet;
330             result(2,1) = -(at(0,0)*at(2,1)-at(2,0)*at(0,1))*invdet;
331             result(2,2) = (at(0,0)*at(1,1)-at(1,0)*at(0,1))*invdet;
332             return result;
333     }
334     return Matrix3X3();
335 }
```

10.57.2.8 float& Matrix3X3::operator()(int i, int j) [inline]

```
285 { return m[3*i+j]; }
```

10.57.2.9 const float& Matrix3X3::operator()(int i, int j) const [inline]

```
286 { return m[3*i+j]; }
```

10.57.2.10 Matrix3X3& Matrix3X3::operator*=(float a) [inline]

```

337
338     for(int i=0;i<9;i++)
339         m[i]*=a;
340     return *this;
341 }
```

10.57.2.11 Matrix3X3& Matrix3X3::operator+=(const Matrix3X3 & b) [inline]

```

343
344     for(int i=0;i<9;i++)
345         m[i]+=b[i];
346     return *this;
347 }
```

10.57.2.12 float& Matrix3X3::operator[](int i) [inline]

```
279 { return m[i]; }
```

10.57.2.13 const float& Matrix3X3::operator[](int i) const [inline]

```
280 { return m[i]; }
```

10.57.2.14 vector3f& Matrix3X3::row(int i) [inline]

```
282 { return *(vector3f*)&m[i*3]; }
```

10.57.2.15 const vector3f& Matrix3X3::row(int i) const [inline]

```
283 { return *(vector3f*)&m[i*3]; }
```

10.57.2.16 static void Matrix3X3::test() [inline], [static]

```

352 {
353     Matrix3X3 t;
354     for (int i=0;i<9;i++)
355         t[i]=i*i;
356     t.Inverse().dbgprint();
357 }
358 }
```

10.57.3 Member Data Documentation

10.57.3.1 float Matrix3X3::m[9]

The documentation for this class was generated from the following file:

- cputrack/utils.h

10.58 MeasureTime Struct Reference

```
#include <gpu_utils.h>
```

Public Member Functions

- `MeasureTime (const char *name)`
- `~MeasureTime ()`

Public Attributes

- `uint64_t freq`
- `uint64_t time`
- `const char * name`

10.58.1 Constructor & Destructor Documentation

10.58.1.1 `MeasureTime::MeasureTime (const char * name) [inline]`

```

163
164     {
165         QueryPerformanceCounter((LARGE_INTEGER*)&time);
166         QueryPerformanceFrequency((LARGE_INTEGER*)&freq);
167         this->name=name;
168     }

```

10.58.1.2 `MeasureTime::~MeasureTime () [inline]`

```

168
169     {
170         uint64_t time1;
171         QueryPerformanceCounter((LARGE_INTEGER*)&time1);
172         double dt = (double)(time1-time) / (double)freq;
173         dbgprintf("%s: Time taken: %f ms\n", name, dt*1000);
174     }

```

10.58.2 Member Data Documentation

10.58.2.1 `uint64_t MeasureTime::freq`

10.58.2.2 `const char* MeasureTime::name`

10.58.2.3 `uint64_t MeasureTime::time`

The documentation for this struct was generated from the following file:

- `cudatrack/gpu_utils.h`

10.59 Threads::Mutex Struct Reference

```
#include <threads.h>
```

Public Member Functions

- `Mutex (const char *name=0)`
- `~Mutex ()`
- `void lock ()`
- `void unlock ()`
- `void msg (const char *m)`

Public Attributes

- `HANDLE h`
- `std::string name`
- `bool trace`
- `int lockCount`

10.59.1 Constructor & Destructor Documentation

10.59.1.1 Threads::Mutex (const char * name = 0) [inline]

```

67             : name(name?name:"") {
68         msg("create");
69         h=CreateMutex(0,FALSE,0);
70         trace=false;
71         lockCount=0;
72     }

```

10.59.1.2 Threads::Mutex::~Mutex () [inline]

```
73 { msg("end"); CloseHandle(h); }
```

10.59.2 Member Function Documentation

10.59.2.1 void Threads::Mutex::lock () [inline]

```

74     {
75         msg("lock");
76         WaitForSingleObject(h, INFINITE);
77         lockCount++;
78     }

```

10.59.2.2 void Threads::Mutex::msg (const char * m) [inline]

```

84     {
85         if(name.length()>0 && trace) {
86             char buf[32];
87             SNPRINTF(buf, sizeof(buf), "mutex %s: %s\n", name.c_str(), m);
88             OutputDebugString(buf);
89         }
90     }

```

10.59.2.3 void Threads::Mutex::unlock() [inline]

```
79             {
80         msg("unlock");
81         lockCount--;
82         ReleaseMutex(h);
83     }
```

10.59.3 Member Data Documentation

10.59.3.1 HANDLE Threads::Mutex::h

10.59.3.2 int Threads::Mutex::lockCount

10.59.3.3 std::string Threads::Mutex::name

10.59.3.4 bool Threads::Mutex::trace

The documentation for this struct was generated from the following file:

- cputrack/[threads.h](#)

10.60 my_error_mgr Struct Reference

Public Attributes

- struct jpeg_error_mgr [pub](#)

10.60.1 Member Data Documentation

10.60.1.1 struct jpeg_error_mgr my_error_mgr::pub

The documentation for this struct was generated from the following file:

- cputrack/[fastjpg.cpp](#)

10.61 outputter::outputModes Struct Reference

Public Attributes

- bool [File](#)
- bool [Console](#)
- bool [Images](#)

10.61.1 Member Data Documentation

10.61.1.1 `bool outputter::outputModes::Console`

10.61.1.2 `bool outputter::outputModes::File`

10.61.1.3 `bool outputter::outputModes::Images`

The documentation for this struct was generated from the following file:

- cputrack-test/[testutils.h](#)

10.62 outputter Class Reference

```
#include <testutils.h>
```

Classes

- struct [outputModes](#)

Public Member Functions

- [outputter](#) (int mode=1)
- [~outputter](#) ()
- void [outputString](#) (std::string out, bool ConsoleOnly=false)
- void [outputImage](#) ([ImageData](#) img, std::string filename="UsedImage")
- template<typename T >
void [outputArray](#) (T *arr, int size)
- void [newFile](#) (std::string filename, const char *mode="a")

Public Attributes

- std::string [folder](#)

Private Member Functions

- void [init](#) (int mode)

Private Attributes

- [outputModes](#) [modes](#)
- FILE * [outputFile](#)

10.62.1 Constructor & Destructor Documentation

10.62.1.1 `outputter::outputter (int mode = 1)`

```
63 { init(mode); }
```

10.62.1.2 `outputter::~outputter ()`

```
65 {
66     if(modes.File && outputFile != NULL)
67         fclose(outputFile);
68 }
```

10.62.2 Member Function Documentation

10.62.2.1 `void outputter::init (int mode) [private]`

```
109 {
110     modes.Console    = (mode & Console) != 0;
111     modes.File      = (mode & Files) != 0;
112     modes.Images    = (mode & Images) != 0;
113
114     outputFile     = NULL;
115
116     if(!modes.Console && !modes.File){
117         modes.Console = true;
118         printf_s("No output mode selected, using console by default.\n");
119     }
120
121     if(modes.File || modes.Images){
122         char date[14];
123         GetFormattedTimeString(date);
124         folder = "D:\\TestImages\\TestOutput\\" + std::string(date) + "\\";
125         if(!CreateDirectory((LPCTSTR)folder.c_str(),NULL)){
126             printf_s("Error creating output folder");
127             throw("");
128         }
129     }
130 }
```

10.62.2.2 `void outputter::newFile (std::string filename, const char * mode = "a")`

```
89 {
90     if(modes.File){
91         if(outputFile)
92             fclose(outputFile);
93         std::string outfile = folder + filename + ".txt";
94         while(fopen(outfile.c_str(),"r")){
95             std::cout << "Output file " << filename << " already exists, please specify new filename:\n";
96             std::cin >> filename;
97             outfile = folder + filename + ".txt";
98         }
99         outputFile = fopen(outfile.c_str(),mode);
100        while(!outputFile){
101            std::cout << "Error creating output file " << outfile << "\n\n";
102            std::cout << "Error " << outfile << " already exists, please specify new file path:\n";
103            std::cin >> outfile;
104            outputFile = fopen(outfile.c_str(),mode);
105        }
106    }
107 }
```

10.62.2.3 template<typename T> void outputter::outputArray (T * arr, int size) [inline]

```

29
30     std::ostringstream out;
31     for(int ii=0;ii<size;ii++){
32         out << "[" << ii << "] : " << arr[ii] << "\n";
33     }
34     outputString(out.str());
35 }
```

10.62.2.4 void outputter::outputImage (ImageData img, std::string filename = "UsedImage")

```

82
83     if(modes.Images){
84         std::string file = folder + filename + ".jpg";
85         FloatToJPEGFile(file.c_str(),img.data,img.w,img.h);
86     }
87 }
```

10.62.2.5 void outputter::outputString (std::string out, bool ConsoleOnly = false)

```

70
71     if(modes.Console || ConsoleOnly){
72         std::cout << out << std::endl;
73     }
74
75     if(modes.File && !ConsoleOnly){
76         if(!outputFile)
77             newFile("OutputFile");
78         fprintf_s(outputFile,"%s\n",out.c_str());
79     }
80 }
```

10.62.3 Member Data Documentation

10.62.3.1 std::string outputter::folder

10.62.3.2 outputModes outputter::modes [private]

10.62.3.3 FILE* outputter::outputFile [private]

The documentation for this class was generated from the following files:

- cputrack-test/testutils.h
- cputrack-test/testutils.cpp

10.63 PathSeparator Struct Reference

```
#include <utils.h>
```

Public Member Functions

- [PathSeparator \(std::string fullPath\)](#)

Public Attributes

- std::string `filename`
- std::string `extension`
- std::string `directory`

10.63.1 Constructor & Destructor Documentation

10.63.1.1 PathSeperator::PathSeperator (std::string *fullpath*)

```

71 {
72     int filenameEnd = fullpath.size();
73     int filenameStart = 0;
74     for (int i = fullpath.size()-1; i>=0; i--) {
75         if (fullpath[i] == '.' && extension.empty()) {
76             extension = fullpath.substr(i+1);
77             filenameEnd = i;
78         }
79         if (fullpath[i] == '/' || fullpath[i] == '\\') {
80             directory = fullpath.substr(0,i+1);
81             filenameStart = i+1;
82             break;
83         }
84     }
85     filename = fullpath.substr(filenameStart, filenameEnd - filenameStart);
86 }
```

10.63.2 Member Data Documentation

10.63.2.1 std::string PathSeperator::directory

10.63.2.2 std::string PathSeperator::extension

10.63.2.3 std::string PathSeperator::filename

The documentation for this struct was generated from the following files:

- cputrack/utils.h
- cputrack/utils.cpp

10.64 pinned_array< T, flags > Class Template Reference

```
#include <gpu_utils.h>
```

Public Member Functions

- `pinned_array ()`
- `~pinned_array ()`
- `pinned_array (size_t n)`
- template<typename TOther , int f>
`pinned_array (const pinned_array< TOther, f > &src)`
- template<typename TOther , int F>
`pinned_array & operator= (const pinned_array< TOther, F > &src)`
- template<typename Iterator >
`pinned_array (Iterator first, Iterator end)`
- template<typename T >
`pinned_array (const device_vec< T > &src)`
- `int size () const`
- `T * begin ()`
- `T * end ()`
- `const T * begin () const`
- `const T * end () const`
- `T * data ()`
- `void free ()`
- `void init (int n)`
- `T & operator[] (int i)`
- `const T & operator[] (int i) const`
- `size_t memsize ()`

Protected Attributes

- `T * d`
- `size_t n`

10.64.1 Constructor & Destructor Documentation

10.64.1.1 template<typename T, int flags = 0> pinned_array< T, flags >::pinned_array () [inline]

```
186
187     d=0;  n=0;
188 }
```

10.64.1.2 template<typename T, int flags = 0> pinned_array< T, flags >::~pinned_array () [inline]

```
189
190     free();
191 }
```

10.64.1.3 template<typename T, int flags = 0> pinned_array< T, flags >::pinned_array (size_t n) [inline]

```
192
193     d=0;  init(n);
194 }
```

10.64.1.4 template<typename T, int flags = 0> template<typename TOther , int f> pinned_array< T, flags >::pinned_array (const pinned_array< TOther,f > & src) [inline]

```
196
197     d=0; init(src.n);
198     for(int k=0;k<src.n;k++)
199         d[k]=src[k];
200 }
```

10.64.1.5 template<typename T, int flags = 0> template<typename Iterator > pinned_array< T, flags >::pinned_array (Iterator first, Iterator end) [inline]

```
209
210     d=0; init(end-first);
211     for (int k = 0; first != end; ++first) {
212         d[k++] = *first;
213     }
214 }
```

10.64.1.6 template<typename T, int flags = 0> template<typename T > pinned_array< T, flags >::pinned_array (const device_vec< T > & src) [inline]

```
216
217     d=0; init(src.size()); src.copyToHost(d, false);
218 }
```

10.64.2 Member Function Documentation

10.64.2.1 template<typename T, int flags = 0> T* pinned_array< T, flags >::begin() [inline]

```
221 { return d; }
```

10.64.2.2 template<typename T, int flags = 0> const T* pinned_array< T, flags >::begin() const [inline]

```
223 { return d; }
```

10.64.2.3 template<typename T, int flags = 0> T* pinned_array< T, flags >::data() [inline]

```
225 { return d; }
```

10.64.2.4 template<typename T, int flags = 0> T* pinned_array< T, flags >::end() [inline]

```
222 { return d+n; }
```

10.64.2.5 template<typename T, int flags = 0> const T* pinned_array< T, flags >::end() const [inline]

```
224 { return d+n; }
```

10.64.2.6 template<typename T, int flags = 0> void pinned_array< T, flags >::free() [inline]

```
226         {
227             cudaFreeHost(d);
228             d=0;n=0;
229 }
```

10.64.2.7 template<typename T, int flags = 0> void pinned_array< T, flags >::init(int n) [inline]

```
230         {
231             if (d) free();
232             this->n = n;
233             if (cudaMallocHost(&d, sizeof(T)*n, flags) != cudaSuccess) {
234                 throw std::bad_alloc(Sprintf("%s init %d elements failed", typeid(*this).name(),
235 n).c_str());
236 }
```

10.64.2.8 template<typename T, int flags = 0> size_t pinned_array< T, flags >::memsize() [inline]

```
239 { return n*sizeof(T); }
```

10.64.2.9 template<typename T, int flags = 0> template<typename TOther , int F> pinned_array& pinned_array< T, flags >::operator=(const pinned_array< TOther, F > & src) [inline]

```
202                                     {
203             if (src.n != n) init(src.n);
204             for(int k=0;k<src.n;k++)
205                 d[k]=src[k];
206             return *this;
207 }
```

10.64.2.10 template<typename T, int flags = 0> T& pinned_array< T, flags >::operator[](int i) [inline]

```
237 { return d[i]; }
```

10.64.2.11 template<typename T, int flags = 0> const T& pinned_array< T, flags >::operator[](int i) const [inline]

```
238 { return d[i]; }
```

10.64.2.12 template<typename T, int flags = 0> int pinned_array< T, flags >::size() const [inline]

```
220 { return n; }
```

10.64.3 Member Data Documentation

10.64.3.1 template<typename T, int flags = 0> T* pinned_array< T, flags >::d [protected]

10.64.3.2 template<typename T, int flags = 0> size_t pinned_array< T, flags >::n [protected]

The documentation for this class was generated from the following file:

- [cudatrack/gpu_utils.h](#)

10.65 BeadFinder::Position Struct Reference

```
#include <BeadFinder.h>
```

Public Member Functions

- [Position \(int x=0, int y=0\)](#)

Public Attributes

- [int x](#)
- [int y](#)

10.65.1 Constructor & Destructor Documentation

10.65.1.1 BeadFinder::Position::Position (int x = 0, int y = 0) [inline]

```
8 : x(x), y(y) {}
```

10.65.2 Member Data Documentation

10.65.2.1 [int BeadFinder::Position::x](#)

10.65.2.2 [int BeadFinder::Position::y](#)

The documentation for this struct was generated from the following file:

- [cputracking/BeadFinder.h](#)

10.66 QI Class Reference

```
#include <QI.h>
```

Classes

- struct [DeviceInstance](#)
- struct [StreamInstance](#)

Public Member Functions

- template<typename TImageSampler>
void [Execute](#) (BaseKernelParams &p, const QTrkComputedConfig &cfg, StreamInstance *s, DeviceInstance *d, device_vec<float3> *initial, device_vec<float3> *output)
- void [InitDevice](#) (DeviceInstance *d, QTrkComputedConfig &cc)
- void [InitStream](#) (StreamInstance *s, QTrkComputedConfig &cc, cudaStream_t stream, int batchSize)
- void [Init](#) (QTrkComputedConfig &cfg, int batchSize)

Private Member Functions

- template<typename TImageSampler>
void [Iterate](#) (BaseKernelParams &p, device_vec<float3> *initial, device_vec<float3> *output, StreamInstance *s, DeviceInstance *d, int angularSteps)
- dim3 [blocks](#) (int njobs)
- dim3 [threads](#) ()

Private Attributes

- QIParams params
- int [qi_FFT_length](#)
- int [batchSize](#)
- int [numThreads](#)

10.66.1 Member Function Documentation

10.66.1.1 dim3 QI::blocks (int njobs) [inline], [private]

```
76 { return dim3((njobs+numThreads-1)/numThreads); }
```

10.66.1.2 template<typename TImageSampler> void QI::Execute (BaseKernelParams & p, const QTrkComputedConfig & cfg, QI::StreamInstance * s, QI::DeviceInstance * d, device_vec<float3> * initial, device_vec<float3> * output)

```
262 {
263     float angsteps = cfg.qi_angstepspq / powf(cfg.qi_angstep_factor, cfg.
264     qi_iterations);
265     for (int a=0;a<cfg.qi_iterations;a++) {
266         device_vec<float3>* dst = a==0 ? initial : output;
267         Iterate< TImageSampler >(p, dst, output, s, d, std::max(
268             MIN_RADPROFILE_SMP_COUNT, (int)angsteps));
269         angsteps *= cfg.qi_angstep_factor;
270     }
}
```

10.66.1.3 void Ql::Init (QTrkComputedConfig & cfg, int batchSize)

```

379 {
380     QIParams& qi = params;
381     qi.trigtablesize = cfg.qi_angstepspq;
382     qi.iterations = cfg.qi_iterations;
383     qi.maxRadius = cfg.qi_maxradius;
384     qi.minRadius = cfg.qi_minradius;
385     qi.radialSteps = cfg.qi_radialsteps;
386
387     cudaDeviceProp prop;
388     cudaGetDeviceProperties(&prop, 0);
389     numThreads = prop.warpSize;
390
391     this->batchSize = batchSize;
392 }
```

10.66.1.4 void Ql::InitDevice (DeviceInstance * d, QTrkComputedConfig & cc)

```

336 {
337     std::vector<float2> qi_radialgrid(cc.qi_angstepspq);
338     for (int i=0;i<cc.qi_angstepspq;i++) {
339         float ang = 0.5f*3.141593f*(i+0.5f)/(float)cc.qi_angstepspq;
340         qi_radialgrid[i]=make_float2(cos(ang), sin(ang));
341     }
342     d->qi_trigtable = qi_radialgrid;
343
344     std::vector<float> rweights = ComputeRadialBinWindow(cc.
345     qi_radialsteps);
346     d->d_radialweights = rweights;
347
348     QIParams dp = params;
349     dp.cos_sin_table = d->qi_trigtable.data;
350
351     cudaMalloc(&d->d_qiparams, sizeof(QIParams));
352     cudaMemcpy(d->d_qiparams, &dp, sizeof(QIParams), cudaMemcpyHostToDevice);
353 }
```

10.66.1.5 void Ql::InitStream (StreamInstance * s, QTrkComputedConfig & cc, cudaStream_t stream, int batchSize)

```

355 {
356     int fftlen = cc.qi_radialsteps*2;
357     s->stream = stream;
358     s->d_quadrants.init(fftlen*batchSize*2);
359     s->d_QIprofiles.init(batchSize*2*fftlen); // (2 axis) * (2 radialsteps) = 8 * nr = 2 * fftlen
360     s->d_QIprofiles_reverse.init(batchSize*2*fftlen);
361     s->d_shiftbuffer.init(fftlen * batchSize);
362
363     // 2* batchSize, since X & Y both need an FFT transform
364     // cufftPlanMany and 1d with batch argument are equivalent in memory usage and speed
365     // Using Many because the batch argument for 1d is strictly speaking deprecated (see cufft.h)
366     cufftResult_t r = cufftPlanMany(&s->fftPlan, 1, &fftlen, 0, 1, fftlen, 0, 1, fftlen, CUFFT_C2C,
367     batchSize*2);
368     //cufftResult_t r = cufftPlan1d(&s->fftPlan, fftlen, CUFFT_C2C, batchSize*2);
369
370     if(r != CUFFT_SUCCESS) {
371         throw std::runtime_error( SPrintf("CUFFT plan creation failed. FFT len: %d. Batchsize: %d\n",
372         fftlen, batchSize*2));
373     }
374     CheckCUDAError(cufftSetCompatibilityMode(s->fftPlan, CUFFT_COMPATIBILITY_NATIVE));
375     CheckCUDAError(cufftSetStream(s->fftPlan, stream));
376
377     this->qi_FFT_length = fftlen;
378 }
```

10.66.1.6 template<typename TImageSampler > void QI::Iterate (BaseKernelParams & p, device_vec<float3> * initial, device_vec<float3> * output, StreamInstance * s, DevicelInstance * d, int angularSteps) [private]

```

274 {
275     if (1) {
276         // This part makes better use of parallelism (computing every quadrant's radial bin in a seperate
277         // thread)
278         dim3 qdrThreads(16, 8, 4);
279         dim3 qdrDim( (p.njobs + qdrThreads.x - 1) / qdrThreads.x, (params.
280         radialSteps + qdrThreads.y - 1) / qdrThreads.y);
281
282         QI_ComputeQuadrants<TImageSampler> <<< qdrDim , qdrThreads, 0, s->stream >>>
283         (p, initial->data, s->d_quadrants.data, d->d_qiparams, angularSteps);
284
285         QI_QuadrantsToProfiles <<< blocks(p.njobs),
286         threads(), 0, s->stream >>>
287         (p, s->d_quadrants.data, s->d_QIprofiles.data, s->d_QIprofiles_reverse.data, d->d_radialweights
288         .data, d->d_qiparams);
289         // DbgOutputVectorToFile("D:\\TestImages\\gpu_qi_prof_new.csv", s->d_quadrants, true);
290     } else {
291         QIPParams dp = params;
292         dp.cos_sin_table = d->qi_trigtable.data;
293
294         QI_ComputeProfile <TImageSampler> <<< blocks(p.njobs),
295         threads(), 0, s->stream >>> (p, initial->data,
296         s->d_quadrants.data, s->d_QIprofiles.data, s->d_QIprofiles_reverse.data, dp, d->d_radialweights
297         .data, angularSteps);
298         // DbgOutputVectorToFile("D:\\TestImages\\gpu_qi_prof_old.csv", s->d_quadrants, true);
299     }
300
301 #ifdef QI_DEBUG
302     DbgCopyResult(s->d_QIprofiles, cmp_gpu_qi_prof);
303 #endif
304
305     cufftComplex* prof = (cufftComplex*)s->d_QIprofiles.data;
306     cufftComplex* revprof = (cufftComplex*)s->d_QIprofiles_reverse.data;
307     CheckCUDAError(cufftExecC2C(s->fftPlan, prof, prof, CUFFT_FORWARD));
308     CheckCUDAError(cufftExecC2C(s->fftPlan, revprof, revprof, CUFFT_FORWARD));
309
310     int nval = qi_FFT_length * 2 * batchSize;
311     int nthread = 256;
312     QI_MultiplyWithConjugate<<< dim3( (nval + nthread - 1)/nthread ), dim3(nthread)
313     , 0, s->stream >>>(nval, prof, revprof);
314     /*
315     Experiment to see the effect of different block sizes.
316     Seems that if the whole parameter space is covered, block sizes don't matter for execution speed.
317
318     dim3 threadsDim = dim3(16, 16);
319     int numBlocks = nval / (threadsDim.x * threadsDim.y);
320     dim3 blocksDim = dim3( sqrt((double)numBlocks)+1, sqrt((double)numBlocks)+1 );
321
322     if(nval > threadsDim.x * threadsDim.y * blocksDim.x * blocksDim.y){
323         printf("\nNot whole space covered: nval > total threads\n");
324         printf("nval: %d, nthread: %d, block (%d,%d), threads (%d,%d)\n", nval, nthread, blocksDim.x,
325         blocksDim.y, threadsDim.x, threadsDim.y);
326     }
327     QI_MultiplyWithConjugate<<< blocksDim, threadsDim, 0, s->stream >>> (nval, prof, revprof);
328     */
329     CheckCUDAError(cufftExecC2C(s->fftPlan, prof, prof, CUFFT_INVERSE));
330
331 #ifdef QI_DEBUG
332     DbgCopyResult(s->d_QIprofiles, cmp_gpu_qi_fft_out);
333 #endif
334
335     float2* d_offsets=0;
336     float pixelsPerProfLen = (params.maxRadius-params.
337     minRadius)/params.radialSteps;
338     dim3 nBlocks=blocks(p.njobs), nThreads=threads();
339     QI_OffsetPositions<<<nBlocks, nThreads, 0, s->stream>>>
340     (p.njobs, initial->data, newpos->data, prof, qi_FFT_length, d_offsets,
341     pixelsPerProfLen, s->d_shiftbuffer.data);
342 }
```

10.66.1.7 dim3 QI::threads() [inline], [private]

```
77 { return dim3(numThreads); }
```

10.66.2 Member Data Documentation

10.66.2.1 int QI::batchSize [private]

10.66.2.2 int QI::numThreads [private]

10.66.2.3 QIParams QI::params [private]

10.66.2.4 int QI::qi_FFT_length [private]

The documentation for this class was generated from the following files:

- cudatrack/QI.h
- cudatrack/QI_impl.h

10.67 QIParams Struct Reference

```
#include <QI.h>
```

Public Attributes

- float minRadius
- float maxRadius
- int radialSteps
- int iterations
- int trigtablesize
- float2 * cos_sin_table

10.67.1 Member Data Documentation

10.67.1.1 float2* QIParams::cos_sin_table

10.67.1.2 int QIParams::iterations

10.67.1.3 float QIParams::maxRadius

10.67.1.4 float QIParams::minRadius

10.67.1.5 int QIParams::radialSteps

10.67.1.6 int QIParams::trigtablesize

The documentation for this struct was generated from the following file:

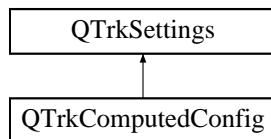
- cudatrack/QI.h

10.68 QTrkComputedConfig Struct Reference

Structure for derived settings computed from base settings in [QTrkSettings](#).

```
#include <qtrk_c_api.h>
```

Inheritance diagram for QTrkComputedConfig:



Public Member Functions

- [QTrkComputedConfig \(\)](#)
- [QTrkComputedConfig \(const QTrkSettings &base\)](#)
- void [Update \(\)](#)
Compute the derived settings.
- void [WriteToLog \(\)](#)
Write all settings to specified log file (Jelmer)
- void [WriteToFile \(\)](#)
Write all settings to specified output file (Jordi, to combine with QTrkSettings.testRun)

Public Attributes

- int [zlut_radialsteps](#)
Number of radial steps to sample on.
- int [zlut_angularsteps](#)
Number of angular steps to sample on.
- float [zlut_maxradius](#)
Max radius in pixels of the sampling circle.
- int [qi_radialsteps](#)
Number of radial steps to sample on.
- int [qi_angstepspq](#)
Number of angular steps to sample on.
- float [qi_maxradius](#)
Max radius in pixels of the sampling circle.

10.68.1 Detailed Description

Structure for derived settings computed from base settings in [QTrkSettings](#).

Compiled without padding to line up with LabVIEW alignment.

Warning

Changing this requires changing of the linked LabVIEW cluster QTrkSettings.ctl.

Note

All settings are always defined, even though only one of the three 2D localization algorithms is used.

10.68.2 Constructor & Destructor Documentation

10.68.2.1 QTrkComputedConfig::QTrkComputedConfig() [inline]

```
171 { }
```

10.68.2.2 QTrkComputedConfig::QTrkComputedConfig(const QTrkSettings & base) [inline]

```
172 { *((QTrkSettings*)this)=base; Update(); }
```

10.68.3 Member Function Documentation

10.68.3.1 void QTrkComputedConfig::Update()

Compute the derived settings.

```
7 {
8     int roi = width / 2;
9
10    zlut_maxradius = roi*zlut_roi_coverage;
11    float zlut_perimeter = 2*3.141593f*zlut_maxradius;
12    zlut_angularsteps = zlut_perimeter*zlut_angular_coverage;
13    zlut_radialsteps = (zlut_maxradius-
14                           zlut_minradius)*zlut_radial_coverage;
15    qi_maxradius = roi*qi_roi_coverage;
16    float qi_perimeter = 2*3.141593f*qi_maxradius;
17    qi_angstepspq = qi_perimeter*qi_angular_coverage/4;
18    qi_radialsteps = (qi_maxradius-qi_minradius)*
19                      qi_radial_coverage;
20    qi_radialsteps = NearestPowerOf2(qi_radialsteps);
21 }
```

10.68.3.2 void QTrkComputedConfig::WriteToFile()

Write all settings to specified output file (Jordi, to combine with [QTrkSettings::testRun](#))

```
57 {
58     std::string path = GetCurrentOutputPath(false);
59     FILE* f = fopen(SPrintf("%s\\UsedSettings.txt",path.c_str()).c_str(),"w+");
60
61 #define WRITEVAR(v) fprintf_s(f,"Setting: %s set to %g\n", #v, (float) v)
62     WRITEVAR(width);
63     WRITEVAR(height);
64     WRITEVAR(numThreads);
65     WRITEVAR(cuda_device);
66     WRITEVAR(com_bgcorrection);
67     WRITEVAR(zlut_minradius);
68     WRITEVAR(zlut_radial_coverage);
69     WRITEVAR(zlut_angular_coverage);
70     WRITEVAR(zlut_roi_coverage);
71     WRITEVAR(qi_iterations);
72     WRITEVAR(qi_minradius);
73     WRITEVAR(qi_radial_coverage);
74     WRITEVAR(qi_angular_coverage);
75     WRITEVAR(qi_roi_coverage);
76     WRITEVAR(qi_angstep_factor);
77     WRITEVAR(xcl_profileLength);
78     WRITEVAR(xcl_profileWidth);
79     WRITEVAR(xcl_iterations);
80     WRITEVAR(gauss2D_iterations);
81     WRITEVAR(gauss2D_sigma);
82     WRITEVAR(zlut_radialsteps);
83     WRITEVAR(zlut_angularsteps);
84     WRITEVAR(zlut_maxradius);
85     WRITEVAR(qi_radialsteps);
86     WRITEVAR(qi_angstepspq);
87     WRITEVAR(qi_maxradius);
88     WRITEVAR(downsample);
89 #undef WRITEVAR
90
91     fclose(f);
92 }
```

10.68.3.3 void QTrkComputedConfig::WriteToLog()

Write all settings to specified log file (Jelmer)

```

24 {
25 #define WRITEVAR(v) dbgprintf("Setting: %s set to %g\n", #v, (float) v)
26     WRITEVAR(width);
27     WRITEVAR(height);
28     WRITEVAR(numThreads);
29     WRITEVAR(cuda_device);
30     WRITEVAR(com_bgcorrection);
31     WRITEVAR(zlut_minradius);
32     WRITEVAR(zlut_radial_coverage);
33     WRITEVAR(zlut_angular_coverage);
34     WRITEVAR(zlut_roi_coverage);
35     WRITEVAR(qi_iterations);
36     WRITEVAR(qi_minradius);
37     WRITEVAR(qi_radial_coverage);
38     WRITEVAR(qi_angular_coverage);
39     WRITEVAR(qi_roi_coverage);
40     WRITEVAR(qi_angstep_factor);
41     WRITEVAR(xcl_profileLength);
42     WRITEVAR(xcl_profileWidth);
43     WRITEVAR(xcl_iterations);
44     WRITEVAR(gauss2D_iterations);
45     WRITEVAR(gauss2D_sigma);
46     WRITEVAR(zlut_radialsteps);
47     WRITEVAR(zlut_angularsteps);
48     WRITEVAR(zlut_maxradius);
49     WRITEVAR(qi_radialsteps);
50     WRITEVAR(qi_angstepspq);
51     WRITEVAR(qi_maxradius);
52     WRITEVAR(downsample);
53 #undef WRITEVAR
54 }
```

10.68.4 Member Data Documentation

10.68.4.1 int QTrkComputedConfig::qi_angstepspq

Number of angular steps to sample on.

10.68.4.2 float QTrkComputedConfig::qi_maxradius

Max radius in pixels of the sampling circle.

10.68.4.3 int QTrkComputedConfig::qi_radialsteps

Number of radial steps to sample on.

10.68.4.4 int QTrkComputedConfig::zlut_angularsteps

Number of angular steps to sample on.

10.68.4.5 float QTrkComputedConfig::zlut_maxradius

Max radius in pixels of the sampling circle.

10.68.4.6 int QTrkComputedConfig::zlut_radialsteps

Number of radial steps to sample on.

The documentation for this struct was generated from the following files:

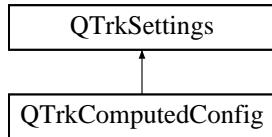
- cputrack/qtrk_c_api.h
- cputrack/QueuedTracker.cpp

10.69 QTrkSettings Struct Reference

Structure for the settings used by the algorithms implemented in [QueuedTracker](#).

```
#include <qtrk_c_api.h>
```

Inheritance diagram for QTrkSettings:



Public Member Functions

- [QTrkSettings \(\)](#)

Public Attributes

- int [width](#)
Width of regions of interest to be handled. Typically equals [QTrkSettings::height](#) (square ROI).
- int [height](#)
Height of regions of interest to be handled. Typically equals [QTrkSettings::width](#) (square ROI).
- int [numThreads](#)
Number of threadsstreams to use. Defaults differ between CPU and GPU implementations.
- int [cuda_device](#)
CUDA only. Flag for device selection.
- float [com_bgcorrection](#)
Background correction factor for COM. Defines the number of standard deviations data needs to be away from the image mean to be taken into account.
- float [zlut_minradius](#)
Distance in pixels from the bead center from which to start sampling profiles. Default 1.0.
- float [zlut_radial_coverage](#)
Sampling points per radial pixel. Default 3.0.
- float [zlut_angular_coverage](#)
Factor of the sampling perimeter to cover with angular sampling steps. Between 0 and 1, default 0.7.
- float [zlut_roi_coverage](#)
*Factor of the ROI to include in sampling. Between 0 and 1, default 1. Maxradius = ROI/2*roi_coverage.*

- int `qi_iterations`
Number of times to run the QI algorithm, sampling around the last found position.
- float `qi_minradius`
Distance in pixels from the bead center from which to start sampling profiles. Default 1.0.
- float `qi_radial_coverage`
Sampling points per radial pixel. Default 3.0.
- float `qi_angular_coverage`
Factor of the sampling perimeter to cover with angular sampling steps. Between 0 and 1, default 0.7.
- float `qi_roi_coverage`
*Factor of the ROI to include in sampling. Between 0 and 1, default 1. Maxradius = ROI/2*roi_coverage.*
- float `qi_angstep_factor`
Factor to reduce angular steps on lower iterations. Default 1.0 (no effect). Increase for faster early iterations but more image noise sensitivity.
- int `xc1_profileLength`
Profile length for the cross correlation.
- int `xc1_profileWidth`
Profile width for the cross correlation.
- int `xc1_iterations`
Number of times to run the cross correlation algorithm.
- int `gauss2D_iterations`
Number of times to run the 2D gaussian algorithm.
- float `gauss2D_sigma`
Standard deviation to use in the 2D gaussian algorithm.
- int `downsample`
Image downsampling factor. Applied before anything else. 0 = original, 1 = 1x (W=W/2,H=H/2).
- bool `testRun`
Flag to run a test run.

10.69.1 Detailed Description

Structure for the settings used by the algorithms implemented in [QueuedTracker](#).

Compiled without padding to line up with LabVIEW alignment.

Warning

Changing this requires changing of the linked LabVIEW cluster `QTrkSettings.ctl`.

Note

All settings are always defined, even though only one of the three 2D localization algorithms is used.

10.69.2 Constructor & Destructor Documentation

10.69.2.1 QTrkSettings::QTrkSettings() [inline]

Set default values on initialization

```

75
76     width = height = 100;
77     numThreads = -1;
78     xcl_profileLength = 128;
79     xcl_profileWidth = 32;
80     xcl_iterations = 2;
81     zlut_minradius = 1.0f;
82     zlut_angular_coverage = 0.7f;
83     zlut_radial_coverage = 3.0f;
84     zlut_roi_coverage = 1.0f;
85     qi_iterations = 5;
86     qi_minradius = 1;
87     qi_angular_coverage = 0.7f;
88     qi_radial_coverage = 3.0f;
89     qi_roi_coverage = 1.0f;
90     qi_angstep_factor = 1.0f;
91     cuda_device = -2;
92     com_bgcorrection = 0.0f;
93     gauss2D_iterations = 6;
94     gauss2D_sigma = 4;
95     downsample = 0;
96     //testRun = false; // CHANGED! Comment/remove when compiling for labview
97 }
```

10.69.3 Member Data Documentation

10.69.3.1 float QTrkSettings::com_bgcorrection

Background correction factor for COM. Defines the number of standard deviations data needs to be away from the image mean to be taken into account.

10.69.3.2 int QTrkSettings::cuda_device

CUDA only. Flag for device selection.

cuda_device ≥ 0	Use as hardware device index.
cuda_device < 0	Use flags below

QTrkCUDA_UseList	-3	Use list defined by SetCUDADevices
QTrkCUDA_UseAll	-2	Use all available devices. Default option.
QTrkCUDA_UseBest	-1	Only use the best device.

10.69.3.3 int QTrkSettings::downsample

Image downsampling factor. Applied before anything else. 0 = original, 1 = 1x (W=W/2,H=H/2).

10.69.3.4 int QTrkSettings::gauss2D_iterations

Number of times to run the 2D gaussian algorithm.

10.69.3.5 float QTrkSettings::gauss2D_sigma

Standard deviation to use in the 2D gaussian algorithm.

10.69.3.6 int QTrkSettings::height

Height of regions of interest to be handled. Typically equals [QTrkSettings::width](#) (square ROI).

10.69.3.7 int QTrkSettings::numThreads

Number of threadsstreams to use. Defaults differ between CPU and GPU implementations.

[CPU]: Default -1 chooses *threads* equal to the number of CPUs available.

[CUDA]: Default -1 chooses 4 *streams* per available device.

10.69.3.8 float QTrkSettings::qi_angstep_factor

Factor to reduce angular steps on lower iterations. Default 1.0 (no effect). Increase for faster early iterations but more image noise sensitivity.

10.69.3.9 float QTrkSettings::qi_angular_coverage

Factor of the sampling perimeter to cover with angular sampling steps. Between 0 and 1, default 0.7.

10.69.3.10 int QTrkSettings::qi_iterations

Number of times to run the QI algorithm, sampling around the last found position.

10.69.3.11 float QTrkSettings::qi_minradius

Distance in pixels from the bead center from which to start sampling profiles. Default 1.0.

10.69.3.12 float QTrkSettings::qi_radial_coverage

Sampling points per radial pixel. Default 3.0.

10.69.3.13 float QTrkSettings::qi_roi_coverage

Factor of the ROI to include in sampling. Between 0 and 1, default 1. Maxradius = ROI/2*roi_coverage.

10.69.3.14 bool QTrkSettings::testRun

Flag to run a test run.

A test run dumps a lot of intermediate data to the disk for algorithm inspection (only QI & ZLUT).

Warning

CHANGED compared to LabVIEW! Comment/remove when compiling for LabVIEW.

Todo Add to LabVIEW cluster.**10.69.3.15 int QTrkSettings::width**

Width of regions of interest to be handled. Typically equals [QTrkSettings::height](#) (square ROI).

10.69.3.16 int QTrkSettings::xc1_iterations

Number of times to run the cross correlation algorithm.

10.69.3.17 int QTrkSettings::xc1_profileLength

Profile length for the cross correlation.

10.69.3.18 int QTrkSettings::xc1_profileWidth

Profile width for the cross correlation.

10.69.3.19 float QTrkSettings::zlut_angular_coverage

Factor of the sampling perimeter to cover with angular sampling steps. Between 0 and 1, default 0.7.

10.69.3.20 float QTrkSettings::zlut_minradius

Distance in pixels from the bead center from which to start sampling profiles. Default 1.0.

10.69.3.21 float QTrkSettings::zlut_radial_coverage

Sampling points per radial pixel. Default 3.0.

10.69.3.22 float QTrkSettings::zlut_roi_coverage

Factor of the ROI to include in sampling. Between 0 and 1, default 1. Maxradius = ROI/2*roi_coverage.

The documentation for this struct was generated from the following file:

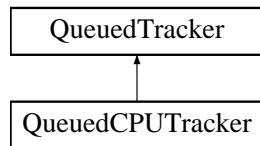
- cputrack/qtrk_c_api.h

10.70 QueuedCPUTracker Class Reference

CPU implementation of the [QueuedTracker](#) interface.

```
#include <QueuedCPUTracker.h>
```

Inheritance diagram for QueuedCPUTracker:



Classes

- struct [Job](#)
- struct [Thread](#)

Public Member Functions

- [QueuedCPUTracker](#) (const [QTrkComputedConfig](#) &cc)
- [~QueuedCPUTracker](#) ()
- void [Start](#) ()
- void [Break](#) (bool pause)
- void [GenerateTestImage](#) (float *dst, float xp, float yp, float z, float photoncount)
- int [NumThreads](#) ()
- void [SetLocalizationMode](#) ([LocMode_t](#) lt) override

Select which algorithm is to be used.
- void [SetRadialZLUT](#) (float *data, int num_zluts, int planes) override
- void [GetRadialZLUT](#) (float *zlut) override
- void [GetRadialZLUTSize](#) (int &count, int &planes, int &rsteps) override
- void [SetRadialWeights](#) (float *rweights) override
- void [SetRadialWeights](#) (std::vector< float > weights)
- void [ScheduleLocalization](#) (void *data, int pitch, [QTRK_PixelDataType](#) pdt, const [LocalizationJob](#) *jobInfo) override

Add a job to the queue to be processed. A job entails running the required algorithms on a single region of interest.
- void [EnableRadialZLUTCompareProfile](#) (bool enabled)
- void [GetRadialZLUTCompareProfile](#) (float *dst)
- void [ZLUTSelfTest](#) ()
- void [BeginLUT](#) (uint flags)

- void [BuildLUT](#) (void *data, int pitch, [QTRK_PixelDataType](#) pdt, int plane, [vector2f](#) *known_pos=0) override
- void [FinalizeLUT](#) () override
- void [GetImageZLUTSize](#) (int *dims)
- void [GetImageZLUT](#) (float *dst)
- bool [SetImageZLUT](#) (float *dst, float *radial_zlut, int *dims)
- void [SetPixelCalibrationImages](#) (float *offset, float *gain) override
- void [SetPixelCalibrationFactors](#) (float offsetFactor, float gainFactor) override
- int [GetQueueLength](#) (int *maxQueueLength=0) override
- int [FetchResults](#) ([LocalizationResult](#) *results, int maxResults) override
- void [ClearResults](#) () override

Clear results.
- void [Flush](#) () override

Stop waiting for more jobs to do, and just process the current batch.
- bool [IsIdle](#) () override
- int [GetResultCount](#) () override
- bool [GetDebugImage](#) (int id, int *w, int *h, float **data)
- [ConfigValueMap](#) [GetConfigValues](#) () override
- void [SetConfigValue](#) (std::string name, std::string value) override
- std::string [GetProfileReport](#) ()
- float * [GetZLUTByIndex](#) (int index)

Private Member Functions

- void [UpdateZLUTs](#) ()
- int [ImageLUTNumBeads](#) ()
- int [ImageLUTWidth](#) ()
- int [ImageLUTHeight](#) ()
- float * [GetImageLUTByIndex](#) (int index, int plane=0)
- void [JobFinished](#) (Job *j)
- Job * [GetNextJob](#) ()
- Job * [AllocateJob](#) ()
- void [AddJob](#) (Job *j)
- void [ProcessJob](#) (Thread *th, Job *j)
- void [SetTrackerImage](#) (CPUTracker *trk, Job *job)
- void [ApplyOffsetGain](#) (CPUTracker *trk, int beadIndex)

Static Private Member Functions

- static void [WorkerThreadMain](#) (void *arg)

Private Attributes

- [LocMode_t](#) localizeMode
- [Threads::Mutex](#) jobs_mutex
- [Threads::Mutex](#) jobs_buffer_mutex
- [Threads::Mutex](#) results_mutex
- std::deque< Job * > jobs
- int jobCount
- std::vector< Job * > jobs_buffer
- std::deque< [LocalizationResult](#) > results
- int resultCount

- int `maxQueueSize`
- int `jobsInProgress`
- `Threads::Mutex gc_mutex`
- float * `calib_gain`
- float * `calib_offset`
- float `gc_gainFactor`
- float `gc_offsetFactor`
- int `downsampleWidth`
- int `downsampleHeight`
- std::vector< `Thread` > `threads`
- float * `zluts`
- float * `zlut_cmpprofiles`
- bool `zlut_enablecmpprof`
- int `zlut_count`
- int `zlut_planes`
- uint `zlut_buildflags`
- std::vector< float > `zcmp`
- std::vector< float > `qi_radialbinweights`
- int `image_lut_dims` [4]
- int `image_lut_nElem_per_bead`
- float * `image_lut`
- float * `image_lut_dz`
- float * `image_lut_dz2`
- bool `quitWork`
- bool `processJobs`
- bool `dbgPrintResults`

Additional Inherited Members

10.70.1 Detailed Description

CPU implementation of the [QueuedTracker](#) interface.

10.70.2 Constructor & Destructor Documentation

10.70.2.1 QueuedCPUTracker::QueuedCPUTracker (const QTrkComputedConfig & cc)

```

97     : jobs_mutex("jobs"), jobs_buffer_mutex("jobs_buffer"),
98     results_mutex("results")
99 {
100     if(0){//cc.testRun)
101         std::string folder = GetCurrentOutputPath();
102         if(GetFileAttributesA(folder.c_str()) & FILE_ATTRIBUTE_DIRECTORY)
103             CreateDirectory((LPCTSTR)folder.c_str(),NULL);
104     }
105     cfg = cc;
106     quitWork = false;
107
108     if (cfg.numThreads < 0) {
109         cfg.numThreads = Threads::GetCPUCount();
110         dbgprintf("Using %d threads\n", cfg.numThreads);
111     }
112
113     maxQueueSize = std::max(2,cfg.numThreads) * 250;
114     jobCount = 0;
115     resultCount = 0;
116
117     zlut_cmpprofiles = 0;

```

```

118     zlut_enablecmpprof = false;
119     zluts = 0;
120     zlut_count = zlut_planes = 0;
121     processJobs = false;
122     jobsInProgress = 0;
123     dbgPrintResults = false;
124
125     qi_radialbinweights = ComputeRadialBinWindow(cfg.
126     qi_radialsteps);
126
127     calib_gain = calib_offset = 0;
128     gc_gainFactor = gc_offsetFactor = 1.0f;
129     localizeMode = LT_OnlyCOM;
130
131     for (int i=0;i<4;i++) image_lut_dims[i]=0;
132     image_lut_nElem_per_bead=0;
133     image_lut = 0;
134     image_lut_dz = image_lut_dz2 = 0;
135
136     downsampleWidth = cfg.width >> cfg.downsample;
137     downsampleHeight = cfg.height >> cfg.downsample;
138
139     Start();
140 }
```

10.70.2.2 QueuedCPUTracker::~QueuedCPUTracker()

```

143 {
144     // wait for threads to finish
145     quitWork = true;
146
147     for (int k=0;k<threads.size();k++) {
148         delete threads[k].mutex;
149         Threads::WaitAndClose(threads[k].thread);
150         delete threads[k].tracker;
151     }
152
153     // free job memory
154     DeleteAllElems(jobs);
155     DeleteAllElems(jobs_buffer);
156
157     delete[] calib_gain;
158     delete[] calib_offset;
159     if (zluts) delete[] zluts;
160     if (zlut_cmpprofiles) delete[] zlut_cmpprofiles;
161     if (image_lut) delete[] image_lut;
162     if (image_lut_dz) delete[] image_lut_dz;
163     if (image_lut_dz2) delete[] image_lut_dz2;
164 }
```

10.70.3 Member Function Documentation

10.70.3.1 void QueuedCPUTracker::AddJob(Job *j) [private]

```

76 {
77     jobs_mutex.lock();
78     jobs.push_back(j);
79     jobCount++;
80     jobs_mutex.unlock();
81 }
```

10.70.3.2 QueuedCPUTracker::Job * QueuedCPUTracker::AllocateJob() [private]

```

63 {
64     QueuedCPUTracker::Job *j;
65     jobs_buffer_mutex.lock();
66     if (!jobs_buffer.empty()) {
67         j = jobs_buffer.back();
68         jobs_buffer.pop_back();
69     } else
70         j = new Job;
71     jobs_buffer_mutex.unlock();
72     return j;
73 }
```

10.70.3.3 void QueuedCPUTracker::ApplyOffsetGain (CPUTracker * *trk*, int *beadIndex*) [private]

```

260 {
261     if (calib_offset || calib_gain) {
262         int index = cfg.width*cfg.height*beadIndex;
263
264         //gc_mutex.lock();
265         //float gf = gc_gainFactor, of = gc_offsetFactor;
266         //gc_mutex.unlock();
267         float gf=1,of=1;
268
269         trk->ApplyOffsetGain(calib_offset ? &
270             calib_offset[index] : 0, calib_gain ? &calib_gain[index] : 0, of, gf);
271     }
272 }
```

10.70.3.4 void QueuedCPUTracker::BeginLUT (uint *flags*) [virtual]

Implements [QueuedTracker](#).

```

581 {
582     zlut_buildflags = flags;
583 }
```

10.70.3.5 void QueuedCPUTracker::Break (bool *pause*)

```

209 {
210     processJobs = !brk;
211 }
```

10.70.3.6 void QueuedCPUTracker::BuildLUT (void * *data*, int *pitch*, QTRK_PixelDataType *pdt*, int *plane*, vector2f * *known_pos* = 0) [override], [virtual]

Implements [QueuedTracker](#).

```

586 {
587     parallel_for(zlut_count,[&] (int i) {
588 //     for(int i=0;i<zlut_count;i++) {
589         CPUTracker trk (cfg.width,cfg.height);
590         void *img_data = (uchar*)data + pitch * cfg.height * i;
591
592         if (pdt == QTrkFloat) {
593             trk.SetImage((float*)img_data, pitch);
594         } else if (pdt == QTrkU8) {
595             trk.SetImage8Bit((uchar*)img_data, pitch);
596         } else {
597             trk.SetImage16Bit((ushort*)img_data,pitch);
598         }
599         ApplyOffsetGain(&trk, i);
600
601         vector2f pos;
602
603         if (known_pos) {
604             pos = known_pos[i];
605         } else {
606             vector2f com = trk.ComputeMeanAndCOM();
607             bool bhit;
608             pos = trk.ComputeQI(com, cfg.qi_iterations, cfg.
609             qi_radialsteps, cfg.qi_angstepspq, cfg.
610             qi_angstep_factor, cfg.qi_minradius, cfg.
611             qi_maxradius, bhit);
612             //dbgprintf("BuildLUT() COMPos: %f,%f, QIPos: x=%f, y=%f\n", com.x,com.y, pos.x, pos.y);
613         }
614         if (zlut_buildflags & BUILDLUT_IMAGELUT) {
```

```

612         int h=ImageLUTHeight(), w=ImageLUTWidth();
613         float* lut_dst = &image_lut[ i * image_lut_nElem_per_bead + w*
614             h* plane ];
614
615         vector2f ilut_scale(1,1);
616         float startx = pos.x - w/2*ilut_scale.x;
617         float starty = pos.y - h/2*ilut_scale.y;
618
619         for (int y=0;y<h;y++) {
620             for (int x=0;x<w;x++) {
621
622                 float px = startx + x*ilut_scale.x;
623                 float py = starty + y*ilut_scale.y;
624
625                 bool outside=false;
626                 float v = Interpolate(trk.srcImage, trk.width, trk.height, px, py, &outside)
627 ;
628                 lut_dst[y*w+x] += v - trk.mean;
629             }
630         }
631
632         float *bead_zlut=GetZLUTByIndex(i);
633         float *tmp = new float[ cfg.zlut_radialsteps];
634
635         if (zlut_buildflags & BUILDLUT_FOURIER){
636             trk.FourierRadialProfile(tmp, cfg.zlut_radialsteps,
637             cfg.zlut_angularsteps, cfg.zlut_minradius,
638             cfg.zlut_maxradius);
639             if (plane==0) {
640                 for (int i=0;i<trk.width*trk.height;i++)
641                     trk.srcImage[i]=sqrtf(trk.srcImage[i]);
642                 trk.SaveImage("freqimg.jpg");
643             }
644             else {
645                 trk.ComputeRadialProfile(tmp, cfg.zlut_radialsteps,
646                 cfg.zlut_angularsteps, cfg.zlut_minradius,
647                 cfg.zlut_maxradius, pos, false);
648             }
649             // WriteArrayAsCSVRow("rlut-test.csv", tmp, cfg.zlut_radialsteps, plane>0);
650             for(int i=0;i<cfg.zlut_radialsteps;i++)
651                 bead_zlut[plane*cfg.zlut_radialsteps+i] += tmp[i];
652             delete[] tmp;
653         }
654     };
655 }
```

10.70.3.7 void QueuedCPUTracker::ClearResults() [override], [virtual]

Clear results.

Implements [QueuedTracker](#).

```

30 {
31     results_mutex.lock();
32     resultCount = 0;
33     results.clear();
34     results_mutex.unlock();
35 }
```

10.70.3.8 void QueuedCPUTracker::EnableRadialZLUTCompareProfile(bool enabled) [virtual]

Implements [QueuedTracker](#).

```

539 {
540     zlut_enablecmpprof=enabled;
541 }
```

10.70.3.9 int QueuedCPUTracker::FetchResults (LocalizationResult * results, int maxResults) [override], [virtual]

Implements [QueuedTracker](#).

```
464 {
465     int numResults = 0;
466     results_mutex.lock();
467     while (numResults < maxResults && !results.empty()) {
468         dstResults[numResults++] = results.back();
469         results.pop_back();
470         resultCount--;
471     }
472     results_mutex.unlock();
473     return numResults;
474 }
```

10.70.3.10 void QueuedCPUTracker::FinalizeLUT () [override], [virtual]

Implements [QueuedTracker](#).

```
654 {
655     // normalize radial LUT?
656
657     if (zluts) {
658         for (int i=0;i<zlut_count*zlut_planes;i++) {
659             // WriteArrayAsCSVRow("finalize-lut.csv", &zluts[cfg.zlut_radialsteps*i], cfg.zlut_radialsteps,
660             i>0);
661             NormalizeRadialProfile(&zluts[cfg.
662             zlut_radialsteps*i], cfg.zlut_radialsteps);
663         }
664     }
665
666     int w = ImageLUTWidth();
667     int h = ImageLUTHeight();
668
669     if (w * h > 0) {
670
671         image_lut_dz = new float [image_lut_nElem_per_bead *
672             ImageLUTNumBeads()];
673         image_lut_dz2 = new float [image_lut_nElem_per_bead *
674             ImageLUTNumBeads()];
675
676         // Compute 1st and 2nd order derivatives
677         for (int i=0;i<image_lut_dims[0];i++) {
678             for (int z=1;z<image_lut_dims[1]-1;z++) {
679                 float *img = &image_lut[ image_lut_nElem_per_bead * i + w*
680                 h*z ]; // current plane
681                 float *imgL = &image_lut[ image_lut_nElem_per_bead * i + w*
682                 h*(z-1) ]; // one plane below
683                 float *imgU = &image_lut[ image_lut_nElem_per_bead * i + w*
684                 h*(z+1) ]; // one plane above
685
686                 float *img_dz = &image_lut_dz[
687                     image_lut_nElem_per_bead * i + w*h*z ];
688                 float *img_dz2 = &image_lut_dz2[
689                     image_lut_nElem_per_bead * i + w*h*z ];
690
691                 // Numerical approx of derivatives..
692                 for (int y=0;y<h;y++) {
693                     for (int x=0;x<w;x++) {
694                         const float h = 1.0f;
695                         img_dz[y*w+x] = 0.5f * ( imgU[y*w+x] - imgL[y*w+x] ) / h;
696                         img_dz2[y*w+x] = (imgU[y*w+x] - 2*img[y*w+x] + imgL[y*w+x]) / (h*h);
697                     }
698                 }
699             }
700         }
701     }
702
703     for (int k=0;k<w*h;k++) {
704         // Top and bottom planes are simply copied from the neighbouring planes
705         image_lut_dz[ image_lut_nElem_per_bead * i + w*h*0 + k]
706         = image_lut_dz[ image_lut_nElem_per_bead * i + w*h*1 + k];
707         image_lut_dz[ image_lut_nElem_per_bead * i + w*h*(
708             image_lut_dims[1]-1) + k] = image_lut_dz[ image_lut_nElem_per_bead * i + w*h*
709             (image_lut_dims[1]-2) + k];
710     }
711 }
```

```

697         image_lut_dz2[ image_lut_nElem_per_bead * i + w*h*0 +
698   k] = image_lut_dz2[ image_lut_nElem_per_bead * i + w*h*1 + k];
699   image_lut_dz2[ image_lut_nElem_per_bead * i + w*h*(
700   image_lut_dims[1]-1) + k] = image_lut_dz2[ image_lut_nElem_per_bead * i +
701   w*h*(image_lut_dims[1]-2) + k];
702 }
```

10.70.3.11 void QueuedCPUTracker::Flush() [inline], [override], [virtual]

Stop waiting for more jobs to do, and just process the current batch.

Implements [QueuedTracker](#).

```
47 { };
```

10.70.3.12 void QueuedCPUTracker::GenerateTestImage (float * dst, float xp, float yp, float z, float photoncount)

```

478 {
479   ImageData img(dst,cfg.width,cfg.height);
480   ::GenerateTestImage(img,xp,yp,z,photoncount);
481 }
```

10.70.3.13 QueuedTracker::ConfigValueMap QueuedCPUTracker::GetConfigValues () [override], [virtual]

Implements [QueuedTracker](#).

```

505 {
506   ConfigValueMap cvm;
507   cvm["trace"] = dbgPrintResults ? "1" : "0";
508   return cvm;
509 }
```

10.70.3.14 bool QueuedCPUTracker::GetDebugImage (int id, int * w, int * h, float ** data) [virtual]

Reimplemented from [QueuedTracker](#).

```

486 {
487   if (id >= 0 && id < threads.size()) {
488     threads[id].lock();
489
490     *w = cfg.width;
491     *h = cfg.height;
492
493     *data = new float [cfg.width*cfg.height];
494     memcpy(*data, threads[id].tracker->GetDebugImage(), sizeof(float)*
495           cfg.width*cfg.height);
496     threads[id].unlock();
497     return true;
498   }
499
500   return false;
501 }
```

10.70.3.15 float* QueuedCPUTracker::GetImageLUTByIndex (int *index*, int *plane* = 0) [inline], [private]

```

113     {
114         return &image_lut [ index * image_lut_nElem_per_bead + plane * (
115             image_lut_dims[2]*image_lut_dims[3]) ];
116     }

```

10.70.3.16 void QueuedCPUTracker::GetImageZLUT (float * *dst*) [virtual]

Reimplemented from [QueuedTracker](#).

```

532 {
533     if (image_lut) {
534         memcpy(dst, image_lut, sizeof(float)*image_lut_nElem_per_bead*
535             image_lut_dims[0]);
536 }

```

10.70.3.17 void QueuedCPUTracker::GetImageZLUTSize (int * *dims*) [virtual]

Reimplemented from [QueuedTracker](#).

```

525 {
526     for (int i=0;i<4;i++)
527         dims[i]=image_lut_dims[i];
528 }

```

10.70.3.18 QueuedCPUTracker::Job * QueuedCPUTracker::GetNextJob () [private]

```

49 {
50     QueuedCPUTracker::Job *j = 0;
51     jobs_mutex.lock();
52     if (!jobs.empty()) {
53         j = jobs.front();
54         jobs.pop_front();
55         jobCount--;
56         jobsInProgress++;
57     }
58     jobs_mutex.unlock();
59     return j;
60 }

```

10.70.3.19 std::string QueuedCPUTracker::GetProfileReport () [inline], [virtual]

Reimplemented from [QueuedTracker](#).

```
56 { return "CPU tracker currently has no profile reporting"; }
```

10.70.3.20 int QueuedCPUTracker::GetQueueLength (int * *maxQueueLength* = 0) [override], [virtual]

Implements [QueuedTracker](#).

```

84 {
85     int jc;
86     jobs_mutex.lock();
87     jc = jobCount + jobsInProgress;
88     jobs_mutex.unlock();
89
90     if (maxQueueLength)
91         *maxQueueLength = this->maxQueueSize;
92
93     return jc;
94 }
```

10.70.3.21 void QueuedCPUTracker::GetRadialZLUT (float * *zlut*) [override], [virtual]

Implements [QueuedTracker](#).

```

431 {
432     int nElem = zlut_planes*cfg.zlut_radialsteps*
        zlut_count;
433     if (nElem>0) {
434         results_mutex.lock();
435         memcpy(zlut, zluts, sizeof(float)* nElem);
436         results_mutex.unlock();
437     }
438 }
```

10.70.3.22 void QueuedCPUTracker::GetRadialZLUTCompareProfile (float * *dst*) [virtual]

Implements [QueuedTracker](#).

```

545 {
546     if (zlut_cmpprofiles) {
547         for (int i=0;i<zlut_count*zlut_planes;i++)
548             dst[i]=zlut_cmpprofiles[i];
549     }
550 }
```

10.70.3.23 void QueuedCPUTracker::GetRadialZLUTSize (int & *count*, int & *planes*, int & *rsteps*) [override], [virtual]

Implements [QueuedTracker](#).

```

423 {
424     count = zlut_count;
425     planes = zlut_planes;
426     rsteps = cfg.zlut_radialsteps;
427 }
```

10.70.3.24 int QueuedCPUTracker::GetResultCount() [override], [virtual]

Implements [QueuedTracker](#).

```
22 {
23     results_mutex.lock();
24     int rc = resultCount;
25     results_mutex.unlock();
26     return rc;
27 }
```

10.70.3.25 float* QueuedCPUTracker::GetZLUTByIndex(int index) [inline]

```
58 { return &zluts[ index * (zlut_planes*cfg.zlut_radialsteps) ]; }
```

10.70.3.26 int QueuedCPUTracker::ImageLUTHeight() [inline], [private]

```
110 { return image_lut_dims[2]; }
```

10.70.3.27 int QueuedCPUTracker::ImageLUTNumBeads() [inline], [private]

```
108 { return image_lut_dims[0]; }
```

10.70.3.28 int QueuedCPUTracker::ImageLUTWidth() [inline], [private]

```
109 { return image_lut_dims[3]; }
```

10.70.3.29 bool QueuedCPUTracker::IsIdle() [override], [virtual]

Implements [QueuedTracker](#).

```
17 {
18     return GetQueueLength() == 0;
19 }
```

10.70.3.30 void QueuedCPUTracker::JobFinished(QueuedCPUTracker::Job *j) [private]

```
38 {
39     jobs_buffer_mutex.lock();
40     jobs_buffer.push_back(j);
41     jobs_buffer_mutex.unlock();
42
43     jobs_mutex.lock();
44     jobsInProgress--;
45     jobs_mutex.unlock();
46 }
```

10.70.3.31 int QueuedCPUTracker::NumThreads() [inline]

```
18 { return cfg.numThreads; }
```

10.70.3.32 void QueuedCPUTracker::ProcessJob(QueuedCPUTracker::Thread *th, Job *j) [private]

```
275 {
276     CPUTracker* trk = th->tracker;
277     th->lock();
278
279     SetTrackerImage(trk, j);
280
281     //FloatToJPEGFile("dbg.jpg", (float*)j->data, cfg.width, cfg.height);
282
283     if (localizeMode & LT_ClearFirstFourPixels) {
284         trk->srcImage[0]=trk->srcImage[1]=trk->srcImage[2]=trk->
285         srcImage[3]=0;
286     }
287
288     ApplyOffsetGain(trk, j->job.zlutIndex);
289 //    dbgprintf("Job: id %d, bead %d\n", j->id, j->zlut);
290
291     LocalizationResult result={};
292     result.job = j->job;
293
294     vector2f com = trk->ComputeMeanAndCOM(cfg.
295 com_bgcorrection);
295     result.imageMean = trk->mean;
296
297     if (_isnan(com.x) || _isnan(com.y))
298         com = vector2f(cfg.width/2, cfg.height/2);
299
300     bool boundaryHit = false;
301
302     if (localizeMode & LT_XCor1D) {
303         result.firstGuess = com;
304         vector2f resultPos = trk->ComputeXCorInterpolated(com,
304 cfg.xcl_iterations, cfg.xcl_profileWidth, boundaryHit);
305         result.pos.x = resultPos.x;
306         result.pos.y = resultPos.y;
307     } else if (localizeMode & LT_QI ) {
308         result.firstGuess = com;
309         vector2f resultPos = trk->ComputeQI(com, cfg.
310 qi_iterations, cfg.qi_radialsteps, cfg.
310 qi_angstepspq, cfg.qi_angstep_factor, cfg.
310 qi_minradius, cfg.qi_maxradius, boundaryHit, &
310 qi_radialbinweights[0]);
311         result.pos.x = resultPos.x;
311         result.pos.y = resultPos.y;
312     } else if (localizeMode & LT_Gaussian2D) {
313         result.firstGuess = com;
314         CPUTracker::Gauss2DResult gr = trk->
314 Compute2DGaussianMLE(com, cfg.gauss2D_iterations,
314 cfg.gauss2D_sigma);
315         vector2f xy = gr.pos;
316         result.pos = vector3f(xy.x,xy.y,0.0f);
317     } else {
318         result.firstGuess.x = result.pos.x = com.x;
319         result.firstGuess.y = result.pos.y = com.y;
320     }
321
322     bool normalizeProfile = (localizeMode & LT_NormalizeProfile)!=0;
323     if(localizeMode & LT_LocalizeZ) {
324         float* prof=ALLOCA_ARRAY(float,cfg.zlut_radialsteps);
325
326         for (int i=0;i< ((localizeMode & LT_ZLUTAlign) ? 5 : 1) ; i++) {
327             if (localizeMode & LT_FourierLUT) {
328                 trk->FourierRadialProfile(prof,cfg.
328 zlut_radialsteps, cfg.zlut_angularsteps, cfg.
328 zlut_minradius, cfg.zlut_maxradius);
329             } else {
330                 trk->ComputeRadialProfile(prof,cfg.
330 zlut_radialsteps, cfg.zlut_angularsteps, cfg.
330 zlut_minradius, cfg.zlut_maxradius, result.pos2D(), false, &boundaryHit
330 , normalizeProfile );
331             }
332
333         float *cmpprof = 0;
334         if (zlut_enablecmpprof)
```

```

336         cmpprof = &zlut_cmpprofiles[j->job.zlutIndex*
337             zlut_planes];
338         if (i > 0) {
339             // update with Quadrant Align
340             result.pos = trk->QuadrantAlign(result.pos, j->job.zlutIndex,
341                 cfg.qi_angstepspq, boundaryHit);
342             result.pos.z = trk->LUTProfileCompare(prof, j->job.zlutIndex, cmpprof,
343                 CPUTracker::LUTProfMaxQuadraticFit, (float*)0, (int*)0, j->job.frame);
344             //dbgprintf("[%d] x=%f, y=%f, z=%f\n", i, result.pos.x, result.pos.y, result.pos.z);
345         }
346         if (localizeMode & LT_LocalizeZWeighted) {
347             result.pos.z = trk->LUTProfileCompareAdjustedWeights(prof,
348                 j->job.zlutIndex, result.pos.z);
349         }
350         if (zlut_bias_correction)
351             result.pos.z = ZLUTBiasCorrection(result.pos.
352                 z, zlut_planes, j->job.zlutIndex);
353     }
354     if(dbgPrintResults)
355         dbgprintf("fr:%d, bead: %d: x=%f, y=%f, z=%f\n",result.job.
356         frame, result.job.zlutIndex, result.pos.x, result.pos.y, result.
357         pos.z);
358     th->unlock();
359     result.error = boundaryHit ? 1 : 0;
360     results_mutex.lock();
361     results.push_back(result);
362     resultCount++;
363     results_mutex.unlock();
364 }
```

10.70.3.33 void QueuedCPUTracker::ScheduleLocalization (void * *data*, int *pitch*, QTRK_PixelDataType *pdt*, const LocalizationJob * *jobInfo*) [override], [virtual]

Add a job to the queue to be processed. A job entails running the required algorithms on a single region of interest.

Parameters

in	<i>data</i>	Pointer to the data. Type specified by [<i>pdt</i>].
in	<i>pitch</i>	Distance in bytes between two successive rows of pixels (e.g. address of (0,0) - address of (0,1)).
in	<i>pdt</i>	Type of [<i>data</i>], specified by QTRK_PixelDataType .
in	<i>jobInfo</i>	Structure with metadata for the ROI to be handled. See LocalizationJob .

Implements [QueuedTracker](#).

```

441 {
442     if (processJobs) {
443         while(maxQueueSize != 0 && GetQueueLength () >=
444             maxQueueSize)
445             Threads::Sleep(5);
446     }
447     Job* j = AllocateJob();
448     int dstPitch = PDT_BytesPerPixel(pdt) * cfg.width;
449     if(!j->data || j->dataType != pdt) {
450         if (j->data) delete[] j->data;
451         j->data = new uchar[dstPitch * cfg.height];
452     }
453     for (int y=0; y<cfg.height; y++)
454         memcpy(&j->data[dstPitch*y], &((uchar*)data)[pitch*y], dstPitch);
455     j->dataType = pdt;
```

```

458     j->job = *jobInfo;
459
460     AddJob(j);
461 }
```

10.70.3.34 void QueuedCPUTracker::SetConfigValue (std::string name, std::string value) [override], [virtual]

Implements [QueuedTracker](#).

```

513 {
514     if (name == "trace")
515         dbgPrintResults = !!atoi(value.c_str());
516 }
```

10.70.3.35 bool QueuedCPUTracker::SetImageZLUT (float * dst, float * radial_zlut, int * dims) [virtual]

Reimplemented from [QueuedTracker](#).

```

554 {
555     if (image_lut) {
556         delete[] image_lut;
557         image_lut=0;
558     }
559
560     for (int i=0;i<4;i++)
561         image_lut_dims[i]=dims[i];
562
563     image_lut_nElem_per_bead = dims[1]*dims[2]*dims[3];
564
565     if (image_lut_nElem_per_bead > 0 && dims[0] > 0) {
566         image_lut = new float [image_lut_nElem_per_bead*
567         image_lut_dims[0]];
567         memset(image_lut, 0, sizeof(float)*image_lut_nElem_per_bead*
568         image_lut_dims[0]);
568     }
569
570     if (src) {
571         memcpy(image_lut, src, sizeof(float)*image_lut_nElem_per_bead*
572         image_lut_dims[0]);
573
574     SetRadialZLUT(radial_zlut, dims[0], dims[1]);
575
576     return true; // returning true indicates this implementation support ImageLUT
577 }
```

10.70.3.36 void QueuedCPUTracker::SetLocalizationMode (LocMode_t locType) [override], [virtual]

Select which algorithm is to be used.

Parameters

in	<i>locType</i>	An integer used as a bitmask for settings based on LocalizeModeEnum .
----	----------------	---

Implements [QueuedTracker](#).

```

519 {
520     while (!IsIdle());
521     localizeMode = lt;
522 }
```

10.70.3.37 void QueuedCPUTracker::SetPixelCalibrationFactors (float *offsetFactor*, float *gainFactor*) [override],
[virtual]

Implements [QueuedTracker](#).

```
200 {
201     gc_mutex.lock();
202     gc_gainFactor = gainFactor;
203     gc_offsetFactor = offsetFactor;
204     gc_mutex.unlock();
205 }
```

10.70.3.38 void QueuedCPUTracker::SetPixelCalibrationImages (float * *offset*, float * *gain*) [override],
[virtual]

Implements [QueuedTracker](#).

```
167 {
168     if (zlut_count > 0) {
169         int nelem = cfg.width*cfg.height*zlut_count;
170
171         if (calib_gain == 0 && gain) {
172             calib_gain = new float[nelem];
173             memcpy(calib_gain, gain, sizeof(float)*nelem);
174         }
175         else if (calib_gain && gain == 0) {
176             delete[] calib_gain;
177             calib_gain = 0;
178         }
179
180         if (calib_offset == 0 && offset) {
181             calib_offset = new float[nelem];
182             memcpy(calib_offset, offset, sizeof(float)*nelem);
183         }
184         else if (calib_offset && offset == 0) {
185             delete[] calib_offset;
186             calib_offset = 0;
187         }
188
189 #ifdef _DEBUG
190     std::string path = GetLocalModulePath();
191     for (int i=0;i<zlut_count;i++) {
192         if(calib_gain) FloatToJPEGFile( Sprintf("%s/gain-bead%d.jpg",
193             path.c_str(), i).c_str(), &calib_gain[cfg.width*cfg.height*i],
194             cfg.width,cfg.height);
195         if(calib_offset) FloatToJPEGFile( Sprintf(
196             "%s/offset-bead%d.jpg", path.c_str(), i).c_str(), &calib_offset[cfg.width*cfg.
197             height*i], cfg.width,cfg.height);
198     }
199 #endif
200 }
```

10.70.3.39 void QueuedCPUTracker::SetRadialWeights (float * *rweights*) [override], [virtual]

Implements [QueuedTracker](#).

```
399 {
400     if (rweights)
401         zcmp.assign(rweights, rweights + cfg.zlut_radialsteps);
402     else
403         zcmp.clear();
404
405     for (int i=0;i<threads.size();i++){
406         threads[i].lock();
407         threads[i].tracker->SetRadialWeights(rweights);
408         threads[i].unlock();
409     }
410 }
```

10.70.3.40 void QueuedCPUTracker::SetRadialWeights (std::vector< float > weights) [inline]

```
26 { SetRadialWeights(&weights[0]); }
```

10.70.3.41 void QueuedCPUTracker::SetRadialZLUT (float * data, int num_zluts, int planes) [override], [virtual]

Implements [QueuedTracker](#).

```
370 {
371 //   jobs_mutex.lock();
372 //   results_mutex.lock();
373 if (zlut_bias_correction)
374     delete zlut_bias_correction;
375
376 if (zluts) delete[] zluts;
377 int res = cfg.zlut_radialsteps;
378 int total = num_zluts*res*planes;
379 if (total > 0) {
380     zluts = new float[planes*res*num_zluts];
381     std::fill(zluts,zluts+(planes*res*num_zluts), 0.0f);
382     zlut_planes = planes;
383     zlut_count = num_zluts;
384     if(data)
385         std::copy(data, data+(planes*res*num_zluts), zluts);
386 }
387 else
388     zluts = 0;
389
390 if (zlut_cmpprofiles) delete[] zlut_cmpprofiles;
391 zlut_cmpprofiles = new float [num_zluts*planes];
392
393 UpdateZLUTs();
394 //   results_mutex.unlock();
395 //   jobs_mutex.unlock();
396 }
```

10.70.3.42 void QueuedCPUTracker::SetTrackerImage (CPUTracker * trk, Job * job) [private]

```
734 {
735     if (j->dataType == QTrkU8) {
736         trk->SetImage8Bit(j->data, cfg.width);
737     } else if (j->dataType == QTrkU16) {
738         trk->SetImage16Bit((ushort*)j->data, cfg.width*2);
739     } else {
740         trk->SetImageFloat((float*)j->data);
741     }
742 }
```

10.70.3.43 void QueuedCPUTracker::Start ()

```
215 {
216     quitWork = false;
217
218     threads.resize(cfg.numThreads);
219     for (int k=0;k<cfg.numThreads;k++) {
220
221         threads[k].mutex = new Threads::Mutex();
222 #if 0
223         threads[k].mutex->name = SPrintf("thread%d", k);
224         threads[k].mutex->trace = true;
225 #endif
226         threads[k].tracker = new CPUTracker(downsampWidth,
227             downsampleHeight, cfg.xcl_profileLength, false); //cfg.testRun);
228         threads[k].manager = this;
229         threads[k].tracker->trackerID = k;
230     }
231
232     for (int k=0;k<threads.size();k++) {
233         threads[k].thread = Threads::Create(
234             WorkerThreadMain, &threads[k]);
235         Threads::SetBackgroundPriority(threads[k].thread, true);
236     }
237     processJobs = true;
238 }
```

10.70.3.44 void QueuedCPUTracker::UpdateZLUTs() [private]

```

414 {
415     for (int i=0;i<threads.size();i++){
416         threads[i].lock();
417         threads[i].tracker->SetRadialZLUT(zluts, zlut_planes,
418                                         cfg.zlut_radialsteps, zlut_count, cfg.
419                                         zlut_minradius, cfg.zlut_maxradius, false, false);
420         threads[i].unlock();
421     }
422 }
```

10.70.3.45 void QueuedCPUTracker::WorkerThreadMain(void * arg) [static], [private]

```

240 {
241     Thread* th = (Thread*)arg;
242     QueuedCPUTracker* this_ = th->manager;
243
244     while (!this_->quitWork) {
245         Job* j = 0;
246         if (this_->processJobs)
247             j = this_->GetNextJob();
248
249         if (j) {
250             this_->ProcessJob(th, j);
251             this_->JobFinished(j);
252         } else {
253             Threads::Sleep(1);
254         }
255     }
256 //dbgprintf("Thread %p ending.\n", arg);
257 }
```

10.70.3.46 void QueuedCPUTracker::ZLUTSelfTest()

```

705 {
706     if (zluts){
707         /*bool compEnabled = zlut_enablecmpprof;
708         if(!compEnabled)
709             EnableRadialZLUTCompareProfile(true);*/
710
711         threads[0].lock();
712         CPUTracker* trk = threads[0].tracker;
713
714         for (int ii=0;ii<zlut_count;ii++) {
715             /*
716             trk.SetRadialZLUT(&zluts[cfg.zlut_radialsteps*zlut_planes*ii], zlut_planes, cfg.zlut_radialsteps, zlut_count, cfg.zlut_minr
717
718             float* curZLUT = GetZLUTByIndex(ii);
719             for(int plane=0;plane<zlut_planes;plane++){
720                 float* cmpprof = new float[zlut_planes];
721                 trk->LUTProfileCompare(&curZLUT[plane*cfg.
722                                         zlut_radialsteps],ii,cmpprof,CPUTracker::LUTProfMaxQuadraticFit
723             );
724                 WriteArrayAsCSVRow("D:\\TestImages\\zlutSelfTest.csv",cmpprof,zlut_planes
725             ,true);
726                 delete[] cmpprof;
727             }
728
729             //EnableRadialZLUTCompareProfile(compEnabled);
730         }
731 }
```

10.70.4 Member Data Documentation

10.70.4.1 `float* QueuedCPUTracker::calib_gain [private]`

10.70.4.2 `float * QueuedCPUTracker::calib_offset [private]`

10.70.4.3 `bool QueuedCPUTracker::dbgPrintResults [private]`

10.70.4.4 `int QueuedCPUTracker::downsampleHeight [private]`

10.70.4.5 `int QueuedCPUTracker::downsampleWidth [private]`

10.70.4.6 `float QueuedCPUTracker::gc_gainFactor [private]`

10.70.4.7 `Threads::Mutex QueuedCPUTracker::gc_mutex [private]`

10.70.4.8 `float QueuedCPUTracker::gc_offsetFactor [private]`

10.70.4.9 `float* QueuedCPUTracker::image_lut [private]`

10.70.4.10 `int QueuedCPUTracker::image_lut_dims[4] [private]`

10.70.4.11 `float * QueuedCPUTracker::image_lut_dz [private]`

10.70.4.12 `float * QueuedCPUTracker::image_lut_dz2 [private]`

10.70.4.13 `int QueuedCPUTracker::image_lut_nElem_per_bead [private]`

10.70.4.14 `int QueuedCPUTracker::jobCount [private]`

10.70.4.15 `std::deque<Job*> QueuedCPUTracker::jobs [private]`

10.70.4.16 `std::vector<Job*> QueuedCPUTracker::jobs_buffer [private]`

10.70.4.17 `Threads::Mutex QueuedCPUTracker::jobs_buffer_mutex [private]`

10.70.4.18 `Threads::Mutex QueuedCPUTracker::jobs_mutex [private]`

10.70.4.19 `int QueuedCPUTracker::jobsInProgress [private]`

10.70.4.20 `LocMode_t QueuedCPUTracker::localizeMode [private]`

10.70.4.21 `int QueuedCPUTracker::maxQueueSize [private]`

10.70.4.22 `bool QueuedCPUTracker::processJobs [private]`

- 10.70.4.23 `std::vector<float> QueuedCPUTracker::qi_radialbinweights` [private]
- 10.70.4.24 `bool QueuedCPUTracker::quitWork` [private]
- 10.70.4.25 `int QueuedCPUTracker::resultCount` [private]
- 10.70.4.26 `std::deque<LocalizationResult> QueuedCPUTracker::results` [private]
- 10.70.4.27 `Threads::Mutex QueuedCPUTracker::results_mutex` [private]
- 10.70.4.28 `std::vector<Thread> QueuedCPUTracker::threads` [private]
- 10.70.4.29 `std::vector<float> QueuedCPUTracker::zcmp` [private]
- 10.70.4.30 `uint QueuedCPUTracker::zlut_buildflags` [private]
- 10.70.4.31 `float* QueuedCPUTracker::zlut_cmpprofiles` [private]
- 10.70.4.32 `int QueuedCPUTracker::zlut_count` [private]
- 10.70.4.33 `bool QueuedCPUTracker::zlut_enablecmpprof` [private]
- 10.70.4.34 `int QueuedCPUTracker::zlut_planes` [private]
- 10.70.4.35 `float* QueuedCPUTracker::zluts` [private]

The documentation for this class was generated from the following files:

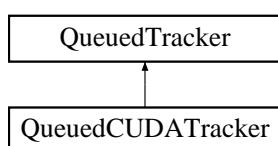
- cputrack/[QueuedCPUTracker.h](#)
- cputrack/[QueuedCPUTracker.cpp](#)

10.71 QueuedCUDATracker Class Reference

CUDA implementation of the [QueuedTracker](#) interface.

```
#include <QueuedCUDATracker.h>
```

Inheritance diagram for QueuedCUDATracker:



Classes

- struct [Device](#)
- struct [KernelProfileTime](#)
- struct [Stream](#)

Public Types

- [typedef Image4DMemory< float > ImageLUT](#)

Public Member Functions

- [QueuedCUDATracker \(const QTrkComputedConfig &cc, int batchSize=-1\)](#)
- [~QueuedCUDATracker \(\)](#)
- void [EnableTextureCache \(bool useTextureCache\)](#)
- void [SetLocalizationMode \(LocMode_t locType\) override](#)
Select which algorithm is to be used.
- void [ScheduleLocalization \(void *data, int pitch, QTRK_PixelDataType pdt, const LocalizationJob *jobInfo\) override](#)
Add a job to the queue to be processed. A job entails running the required algorithms on a single region of interest.
- void [ClearResults \(\) override](#)
Clear results.
- void [SetRadialZLUT \(float *data, int numLUTs, int planes\) override](#)
- void [SetRadialWeights \(float *zcmp\) override](#)
- void [GetRadialZLUT \(float *data\) override](#)
- void [GetRadialZLUTSize \(int &count, int &planes, int &radialSteps\) override](#)
- int [FetchResults \(LocalizationResult *results, int maxResults\) override](#)
- void [BeginLUT \(uint flags\) override](#)
- void [BuildLUT \(void *data, int pitch, QTRK_PixelDataType pdt, int plane, vector2f *known_pos=0\) override](#)
- void [FinalizeLUT \(\) override](#)
- void [EnableRadialZLUTCompareProfile \(bool enabled\)](#)
- void [GetRadialZLUTCompareProfile \(float *dst\)](#)
- std::string [GetProfileReport \(\) override](#)
- void [Flush \(\) override](#)
Stop waiting for more jobs to do, and just process the current batch.
- int [GetQueueLength \(int *maxQueueLen\) override](#)
- bool [IsIdle \(\) override](#)
- int [GetResultCount \(\) override](#)
- void [SetPixelCalibrationImages \(float *offset, float *gain\) override](#)
- void [SetPixelCalibrationFactors \(float offsetFactor, float gainFactor\) override](#)
- [ConfigValueMap GetConfigValues \(\) override](#)
- void [SetConfigValue \(std::string name, std::string value\) override](#)

Public Attributes

- [KernelProfileTime time](#)
- [KernelProfileTime cpu_time](#)
- int [batchesDone](#)
- std::string [deviceReport](#)

Protected Member Functions

- dim3 `blocks` (int workItems)
- dim3 `blocks` ()
- dim3 `threads` ()
- void `SchedulingThreadMain` ()
- template<typename TImageSampler >
void `ExecuteBatch` (Stream *s)
- Stream * `GetReadyStream` ()
- void `InitializeDeviceList` ()
- Stream * `CreateStream` (Device *device, int streamIndex)
- void `CopyStreamResults` (Stream *s)
- void `StreamUpdateZLUTSize` (Stream *s)
- void `CPU_ApplyOffsetGain` (CPUTracker *trk, int beadIndex)

Static Protected Member Functions

- static void `SchedulingThreadEntryPoint` (void *param)

Protected Attributes

- int `numThreads`
- int `batchSize`

Amount of images to be sent at once per stream.
- std::vector< Stream * > `streams`
- std::list< LocalizationResult > `results`
- int `resultCount`
- LocMode_t `localizeMode`
- Threads::Mutex `resultMutex`
- Threads::Mutex `jobQueueMutex`
- std::vector< Device * > `devices`
- bool `useTextureCache`
- float `gc_offsetFactor`
- float `gc_gainFactor`
- Threads::Mutex `gc_mutex`
- std::vector< float > `gc_offset`
- std::vector< float > `gc_gain`
- uint `zlut_build_flags`

cpu size buffers are needed because BuildLUT uses CPU tracking
- QI `qi`
- QI `qalign`
- cudaDeviceProp `deviceProp`
- Threads::Handle * `schedulingThread`
- Atomic< bool > `quitScheduler`

10.71.1 Detailed Description

CUDA implementation of the `QueuedTracker` interface.

10.71.2 Member Typedef Documentation

10.71.2.1 `typedef Image4DMemory<float> QueuedCUDATracker::ImageLUT`

10.71.3 Constructor & Destructor Documentation

10.71.3.1 `QueuedCUDATracker::QueuedCUDATracker(const QTrkComputedConfig & cc, int batchSize = -1)`

```

140      : resultMutex("result"), jobQueueMutex("jobqueue")
141  {
142      cfg = cc;
143
144      InitializeDeviceList();
145
146      // We take numThreads to be the number of CUDA streams
147      if (cfg.numThreads < 1) {
148          cfg.numThreads = devices.size()*4;
149      }
150      int numStreams = cfg.numThreads;
151
152      cudaGetDeviceProperties(&deviceProp, devices[0]->index);
153
154      //if (deviceProp.kernelExecTimeoutEnabled)
155      //    throw std::runtime_error(Sprintf("CUDA Tracker init error: CUDA Kernel execution timeout is enabled
156      //for %s. Disable WDDM Time-out Detection and Recovery (TDR) in the nVidia NSight Monitor before running this
157      //code", deviceProp.name));
158      if (deviceProp.major < 2)
159          throw std::runtime_error("CUDA Tracker init error: GPU not supported, capability < 2.0");
160
161      numThreads = deviceProp.warpSize;
162
163      if(batchSize<0) batchSize = (int)(deviceProp.maxTexture2D[1]/
164      cfg.height * 0.99f);
165      //while (batchSize * cfg.height > deviceProp.maxTexture2D[1]) {
166      //    batchSize/=2;
167      //}
168      this->batchSize = batchSize;
169
170      dbgprintf("CUDA Hardware: %s. \n# of CUDA processors: %d. Using %d streams\n",
171      deviceProp.name, deviceProp.multiProcessorCount, numStreams);
172      dbgprintf("Warp size: %d. Max threads: %d, Batch size: %d\n",
173      deviceProp.warpSize, deviceProp.maxThreadsPerBlock,
174      batchSize);
175      // dbgprintf("Mem: %u MB. Per block: %u B\n", deviceProp.totalGlobalMem/1024/1024,
176      deviceProp.sharedMemPerBlock); // Total memory available in Mbytes
177
178      qi.Init(cfg, batchSize);
179
180      std::vector<float2> zlut_radialgrid(cfg.zlut_angularsteps);
181      for (int i=0;i<cfg.zlut_angularsteps;i++) {
182          float ang = 2*3.141593f*i/(float)cfg.zlut_angularsteps;
183          zlut_radialgrid[i]=make_float2(cos(ang),sin(ang));
184      }
185
186      for (uint i=0;i<devices.size();i++) {
187          Device* d = devices[i];
188          cudaSetDevice(d->index);
189          qi.InitDevice(&d->qi_instance, cfg);
190          d->zlut_trigtable = zlut_radialgrid;
191      }
192
193      dbgprintf("\n");
194      outputTotalGPUMemUse("pre-streams");
195
196      streams.reserve(numStreams);
197      try {
198          for (int i=0;i<numStreams;i++)
199              streams.push_back( CreateStream( devices[i%  
devices.size()], i ) );
200      }
201      catch(...) {
202          DeleteAllElems(streams);
203          throw std::runtime_error("CUDA Tracker init error: Failed to create GPU streams.");
204      }
205
206      dbgprintf("\n");
207      streams[0]->OutputMemoryUse();
208      dbgprintf("\n");
209      outputTotalGPUMemUse("post-streams");
210      dbgprintf("\n");

```

```

204     batchesDone = 0;
205     useTextureCache = true;
206     resultCount = 0;
207     zlut_build_flags=0;
208
209     quitScheduler = false;
210     schedulingThread = Threads::Create(
211         SchedulingThreadEntryPoint, this);
212
213     gc_offsetFactor = gc_gainFactor = 1.0f;
214     localizeMode = LT_OnlyCOM;
215
216     ForceCUDAKernelsToLoad <<< dim3(),dim3() >>> ();
217 }
```

10.71.3.2 QueuedCUDATracker::~QueuedCUDATracker()

```

220 {
221     quitScheduler = true;
222     Threads::WaitAndClose(schedulingThread);
223
224     DeleteAllElems(streams);
225     DeleteAllElems(devices);
226 }
```

10.71.4 Member Function Documentation

10.71.4.1 void QueuedCUDATracker::BeginLUT(uint flags) [override], [virtual]

Implements [QueuedTracker](#).

```

469 {
470     zlut_build_flags = flags;
471 }
```

10.71.4.2 dim3 QueuedCUDATracker::blocks(int workItems) [inline], [protected]

```
188 { return dim3((workItems+numThreads-1)/numThreads); }
```

10.71.4.3 dim3 QueuedCUDATracker::blocks() [inline], [protected]

```
189 { return dim3((batchSize+numThreads-1)/numThreads); }
```

10.71.4.4 void QueuedCUDATracker::BuildLUT (void * *data*, int *pitch*, QTRK_PixelDataType *pdt*, int *plane*, vector2f * *known_pos* = 0) [override], [virtual]

Implements [QueuedTracker](#).

```

474 {
475     // Copy to image
476     Device* d = streams[0]->device;
477     cudaSetDevice(d->index);
478
479     int nbeads = d->radial_zlut.count; // jobcount
480     std::vector<vector2f> positions(nbeads);
481
482     float *profiles = new float [nbeads * cfg.zlut_radialsteps];
483     memset(profiles, 0, sizeof(float) * nbeads * cfg.zlut_radialsteps);
484
485     // Should be fast enough for zlut building
486     parallel_for(nbeads, [&] (int i) {
487         CPUTracker trk (cfg.width,cfg.height);
488         void *img_data = (uchar*)data + pitch * cfg.height * i;
489
490         if (pdt == QTrkFloat)
491             trk.SetImage((float*)img_data, pitch);
492         else if (pdt == QTrkU8)
493             trk.SetImage8Bit((uchar*)img_data, pitch);
494         else
495             trk.SetImage16Bit((ushort*)img_data,pitch);
496
497         CPU_ApplyOffsetGain(&trk, i);
498
499         if(known_pos)
500             positions[i] = known_pos[i];
501         else {
502             vector2f com = trk.ComputeMeanAndCOM();
503             bool bhit;
504             positions[i] = trk.ComputeQI(com, cfg.qi_iterations,
505                                         cfg.qi_radialsteps, cfg.qi_angsteppq, cfg.
506                                         qi_angstep_factor, cfg.qi_minradius, cfg.
507                                         qi_maxradius, bhit);
508             trk.ComputeRadialProfile(&profiles[i * cfg.zlut_radialsteps],
509                                     cfg.zlut_radialsteps, cfg.zlut_angularsteps,
510                                     cfg.zlut_minradius, cfg.zlut_maxradius, positions[i], false, 0, true);
511         }
512     });
513
514     // add to device 0 LUT
515     device_vec<float> d_profiles (nbeads * cfg.
516     zlut_radialsteps);
517     d_profiles.copyToDevice(profiles, nbeads * cfg.zlut_radialsteps);
518     delete[] profiles;
519
520     dim3 numThreads(4, 64);
521     dim3 numBlocks( (nbeads + numThreads.x - 1) / numThreads.x, (
522     cfg.zlut_radialsteps + numThreads.y - 1) /
523     numThreads.y );
524     AddProfilesToZLUT<<< numBlocks, numThreads, 0,
525     streams[0]->stream >>> (d_profiles.data, nbeads, cfg.zlut_radialsteps, plane, d->
526     radial_zlut);
527     cudaStreamSynchronize(streams[0]->stream);
528 }
```

10.71.4.5 void QueuedCUDATracker::ClearResults () [override], [virtual]

Clear results.

Implements [QueuedTracker](#).

```

886 {
887     resultMutex.lock();
888     results.clear();
889     resultCount=0;
890     resultMutex.unlock();
891 }
```

10.71.4.6 void QueuedCUDATracker::CopyStreamResults (Stream * s) [protected]

```

699 {
700     resultMutex.lock();
701     for (int a=0;a<s->JobCount();a++) {
702         LocalizationJob& j = s->jobs[a];
703         LocalizationResult r;
704         r.job = j;
705         r.firstGuess = vector2f( s->com[a].x, s->com[a].y );
706         r.pos = vector3f( s->results[a].x , s->results[a].y , s->results[a].z );
707         r.imageMean = s->imgMeans[a];
708         r.pos.z = ZLUTBiasCorrection(s->results[a].z,
709             devices[0]->radial_zlut.h, j.zlutIndex);
710         results.push_back(r);
711     }
712     resultCount+=s->JobCount();
713     resultMutex.unlock();
714     // Update times
715     float q1, com, imagecopy, zcomp, getResults;
716     cudaEventElapsedTime(&imagecopy, s->batchStart, s->imageCopyDone);
717     cudaEventElapsedTime(&com, s->imageCopyDone, s->comDone);
718     cudaEventElapsedTime(&q1, s->comDone, s->q1Done);
719     cudaEventElapsedTime(&zcomp, s->q1Done, s->zcomputeDone);
720     cudaEventElapsedTime(&getResults, s->zcomputeDone, s->localizationDone);
721     time.com += com;
722     time.q1 += q1;
723     time.imageCopy += imagecopy;
724     time.zcompute += zcomp;
725     time.getResults += getResults;
726     batchesDone++;
727 }
```

10.71.4.7 void QueuedCUDATracker::CPU_ApplyOffsetGain (CPUTracker * trk, int beadIndex) [protected]

```

764 {
765     if (!gc_offset.empty() || !gc_gain.empty()) {
766         int index = cfg.width*cfg.height*beadIndex;
767
768         gc_mutex.lock();
769         float gf = gc_gainFactor, of = gc_offsetFactor;
770         gc_mutex.unlock();
771
772         trk->ApplyOffsetGain(gc_offset.empty() ? 0: &
773             gc_offset[index] , gc_gain.empty() ? 0: &gc_gain[index], of, gf);
774     }
775 }
```

10.71.4.8 QueuedCUDATracker::Stream * QueuedCUDATracker::CreateStream (Device * device, int streamIndex) [protected]

```

334 {
335     Stream* s = new Stream(streamIndex);
336
337     try {
338         s->device = device;
339         cudaSetDevice(device->index);
340         cudaStreamCreate(&s->stream);
341
342         s->images = cudaImageListf::alloc(cfg.width,
343             cfg.height, batchSize);
344         s->images.allocateHostImageBuffer(s->hostImageBuf);
345
346         s->jobs.reserve(batchSize);
347         s->results.init(batchSize);
348         s->com.init(batchSize);
349         s->d_com.init(batchSize);
350         s->d_resultpos.init(batchSize);
351         s->results.init(batchSize);
352         s->locParams.init(batchSize);
353         s->imgMeans.init(batchSize);
354         s->d_imgmeans.init(batchSize);
355         s->d_locParams.init(batchSize);
356         s->d_radialprofiles.init(cfg.zlut_radialsteps*
```

```

batchSize);
356     qi.InitStream(&s->qi_instance, cfg, s->stream, batchSize);
358     qalign.InitStream(&s->qalign_instance, cfg, s->stream,
batchSize);
359     cudaEventCreate(&s->localizationDone);
360     cudaEventCreate(&s->comDone);
361     cudaEventCreate(&s->imageCopyDone);
362     cudaEventCreate(&s->zcomputeDone);
363     cudaEventCreate(&s->qiDone);
364     cudaEventCreate(&s->imapDone);
365     cudaEventCreate(&s->batchStart);
366 } catch (...) {
367     delete s;
368     throw;
369 }
370 }
371 return s;
372 }
```

10.71.4.9 void QueuedCUDATracker::EnableRadialZLUTCompareProfile (bool *enabled*) [inline], [virtual]

Implements [QueuedTracker](#).

```
87 {}
```

10.71.4.10 void QueuedCUDATracker::EnableTextureCache (bool *useTextureCache*) [inline]

```
70 { this->useTextureCache=useTextureCache; }
```

10.71.4.11 template<typename TImageSampler> void QueuedCUDATracker::ExecuteBatch (Stream * *s*) [protected]

```

587 {
588     if (s->JobCount ()==0)
589         return;
//dbgprintf("Sending %d images to GPU stream %p...\n", s->jobCount, s->stream);
591     Device *d = s->device;
592     cudaSetDevice(d->index);
593
594     BaseKernelParams kp;
595     kp.imgmeans = s->d_imgmeans.data;
596     kp.images = s->images;
597     kp.njobs = s->JobCount ();
598     kp.locParams = s->d_locParams.data;
599
600     cudaEventRecord(s->batchStart, s->stream);
601 //    dbgprintf("copying %d jobs to gpu\n", s->JobCount ());
602 //    s->d_locParams.copyToDevice(s->locParams.data(), s->JobCount (), true, s->stream);
603
604     {ScopedCPUProfiler p(&cpu_time.imageCopy);
605      s->images.copyToDevice(s->hostImageBuf.data(), true, s->stream);
606    }
607
608     if (!d->calib_gain.isEmpty() || !d->calib_offset.isEmpty()) {
609         dim3 numThreads(16, 16, 2);
610         dim3 numBlocks((cfg.width + numThreads.x - 1) /
numThreads.x,
611                         (cfg.height + numThreads.y - 1) / numThreads.y,
612                         (s->JobCount () + numThreads.z - 1) / numThreads.z);
613
614         gc_mutex.lock();
615         float of = gc_offsetFactor, gf = gc_gainFactor;
616         gc_mutex.unlock();
617
618         ApplyOffsetGain <<< numBlocks, numThreads, 0, s->stream >>>
619             (kp, s->device->calib_gain, s->device->calib_offset, gf, of);
620     }
621 }
```

```

622     cudaEventRecord(s->imageCopyDone, s->stream);
624
625     TImageSampler::BindTexture(s->images);
626     { ScopedCPUProfiler p(&cpu_time.com);
627       BgCorrectedCOM<TImageSampler> <<< blocks(s->JobCount()), threads(), 0, s->stream >>>
628       (s->JobCount(), s->images, s->d_com.data, cfg.com_bgcorrection, s->
d_imgmeans.data);
629       checksum(s->d_com.data, 1, s->JobCount(), "com");
630     }
631     cudaEventRecord(s->comDone, s->stream);
632
633 // DbgOutputVectorToFile("D:\\TestImages\\imgmeans.csv", s->d_imgmeans, false);
634
635     device_vec<float3> *curpos = &s->d_com;
636     if (s->localizeFlags & LT_QI) {
637       ScopedCPUProfiler p(&cpu_time.qi);
638       qi.Execute<TImageSampler> (kp, cfg, &s->qi_instance, &s->device->qi_instance, &s->
d_com, &s->d_resultpos);
639       curpos = &s->d_resultpos;
640     }
641
642     if (s->localizeFlags & LT_Gaussian2D) {
643       G2MLE_Compute<TImageSampler> <<< blocks(s->JobCount()), threads(), 0, s->stream >>>
644       (kp, cfg.gauss2D_sigma, cfg.gauss2D_iterations, s->d_com.
data, s->d_resultpos.data, 0, 0);
645       curpos = &s->d_resultpos;
646     }
647
648     cudaEventRecord(s->qiDone, s->stream);
649
650     int numZIterations = (s->localizeFlags & LT_ZLUTAlign) ? 3 : 1;
651     for (int i=0;i<numZIterations;i++) {
652       {ScopedCPUProfiler p(&cpu_time.zcompute);
653       ZLUTParams zp;
654       zp.planes = d->radial_zlut.h;
655       zp.angularSteps = cfg.zlut_angularsteps;
656       zp.maxRadius = cfg.zlut_maxradius;
657       zp.minRadius = cfg.zlut_minradius;
658       zp.img = d->radial_zlut;
659       zp.trigtable = d->zlut_trigtable.data;
660       zp.zcmpwindow = d->zcompareWindow.data;
661
662       // Compute radial profiles
663       if (s->localizeFlags & LT_LocalizeZ) {
664         dim3 numThreads(16, 16);
665         dim3 numBlocks( (s->JobCount() + numThreads.x - 1) /
numThreads.x,
666                         (cfg.zlut_radialsteps + numThreads.y - 1) /
numThreads.y);
667         ZLUT_RadialProfileKernel<TImageSampler> <<< numBlocks , numThreads, 0, s->stream >>>
668           (s->JobCount(), s->images, zp, curpos->data, s->d_radialprofiles.data, s->d_imgmeans.
data);
669         ZLUT_NormalizeProfiles<<< blocks(s->JobCount()),
threads(), 0, s->stream >>> (s->JobCount(), zp, s->d_radialprofiles.data);
670       }
671       // Compute Z
672       if (s->localizeFlags & LT_LocalizeZ) {
673         dim3 numThreads(8, 16);
674         ZLUT_ComputeProfileMatchScores <<< dim3( (s->JobCount() +
numThreads.x - 1) / numThreads.x, (zp.planes + numThreads.y - 1) / numThreads.y), numThreads, 0, s->stream
>>>
675           (s->JobCount(), zp, s->d_radialprofiles.data, s->d_zlutzcmplscores.data, s->d_locParams.data)
;
676         ZLUT_ComputeZ <<< blocks(s->JobCount()), threads(), 0, s->stream >>>
677           (s->JobCount(), zp, curpos->data, s->d_zlutzcmplscores.data);
678       }
679
680       if (i>0) {
681         ScopedCPUProfiler p(&cpu_time.zlutAlign);
682         qalign.Execute<TImageSampler> (kp, cfg, &s->qalign_instance, &s->device->
qalign_instance, &s->d_resultpos, &s->d_resultpos);
683     }
684
685     TImageSampler::UnbindTexture(s->images);
686     cudaEventRecord(s->zcomputeDone, s->stream);
687
688     { ScopedCPUProfiler p(&cpu_time.getResults);
689       s->d_com.copyToHost(s->com.data(), true, s->stream);
690       curpos->copyToHost(s->results.data(), true, s->stream);
691       s->d_imgmeans.copyToHost(s->imgMeans.data(), true, s->stream);
692     }
693
694 // Make sure we can query the all done signal
695     cudaEventRecord(s->localizationDone, s->stream);
696   }

```

10.71.4.12 int QueuedCUDATracker::FetchResults (LocalizationResult * results, int maxResults) [override], [virtual]

Implements [QueuedTracker](#).

```

730 {
731     resultMutex.lock();
732     int numResults = 0;
733     while (numResults < maxResults && !results.empty()) {
734         dstResults[numResults++] = results.front();
735         results.pop_front();
736         resultCount--;
737     }
738     resultMutex.unlock();
739     return numResults;
740 }
```

10.71.4.13 void QueuedCUDATracker::FinalizeLUT() [override], [virtual]

Implements [QueuedTracker](#).

```

521 {
522     Device* srcl = devices[0];
523     cudaSetDevice(srcl->index);
524
525     cudaImageListf& src = srcl->radial_zlut;
526     float *tmp = new float [src.w*src.h*src.count];
527     srcl->radial_zlut.copyToHost(tmp);
528
529     for (int i=0;i<src.h*src.count;i++){
530         NormalizeRadialProfile(&tmp[src.w*i],src.w);
531     }
532
533     SetRadialZLUT(tmp, src.count, src.h);
534
535     for(int bead = 0; bead < src.count; bead++){
536         for(int step = 0; step < cfg.zlut_radialsteps; step++){
537             dbgprintf("%f ",tmp[(bead * cfg.zlut_radialsteps)+step]);
538         }
539     }
540     dbgprintf("\n"); /* */
541
542     delete[] tmp;
543 }
```

10.71.4.14 void QueuedCUDATracker::Flush() [override], [virtual]

Stop waiting for more jobs to do, and just process the current batch.

Implements [QueuedTracker](#).

```

546 {
547     jobQueueMutex.lock();
548     for (uint i=0;i<streams.size();i++) {
549         if(streams[i]->JobCount()>0 && streams[i]->state != Stream::StreamExecuting)
550             streams[i]->state = Stream::StreamPendingExec;
551     }
552     jobQueueMutex.unlock();
553 }
```

10.71.4.15 `QueuedCUDATracker::ConfigValueMap QueuedCUDATracker::GetConfigValues() [override], [virtual]`

Implements [QueuedTracker](#).

```
906 {
907     ConfigValueMap cvm;
908     cvm["use_texturecache"] = useTextureCache ? "1" : "0";
909     return cvm;
910 }
```

10.71.4.16 `std::string QueuedCUDATracker::GetProfileReport() [override], [virtual]`

Reimplemented from [QueuedTracker](#).

```
894 {
895     float f = 1.0f/batchesDone;
896
897     return deviceReport + "Time profiling: [GPU], [CPU] \n" +
898         SPrintf("%d batches done of size %d, on %d streams", batchesDone,
899         batchSize, streams.size()) + "\n" +
900         SPrintf("Image copying: %.2f,\t%.2f ms\n", time.imageCopy*f,
901         cpu_time.imageCopy*f) +
902         SPrintf("QI:           %.2f,\t%.2f ms\n", time.qi*f,
903         cpu_time.qi*f) +
904         SPrintf("COM:          %.2f,\t%.2f ms\n", time.com*f,
905         cpu_time.com*f) +
906         SPrintf("Z Computing:   %.2f,\t%.2f ms\n", time.zcompute*f,
907         cpu_time.zcompute*f);
908 }
```

10.71.4.17 `int QueuedCUDATracker::GetQueueLength(int * maxQueueLen) [override], [virtual]`

Implements [QueuedTracker](#).

```
406 {
407     jobQueueMutex.lock();
408     int qlen = 0;
409     for (uint a=0;a<streams.size();a++) {
410         qlen += streams[a]->JobCount();
411     }
412     jobQueueMutex.unlock();
413
414     if (maxQueueLen) {
415         *maxQueueLen = streams.size()*batchSize;
416     }
417
418     return qlen;
419 }
```

10.71.4.18 `void QueuedCUDATracker::GetRadialZLUT(float * data) [override], [virtual]`

Implements [QueuedTracker](#).

```
859 {
860     cudaImageListf* zlut = &devices[0]->radial_zlut;
861
862     if (zlut->data) {
863         for (int i=0;i<zlut->count;i++) {
864             float* img = &data[i*cfg.zlut_radialsteps*zlut->h];
865             zlut->copyImageToHost(i, img);
866         }
867     }
868 }
```

10.71.4.19 void QueuedCUDATracker::GetRadialZLUTCompareProfile (float * dst) [inline], [virtual]

Implements [QueuedTracker](#).

```
88 {} // dst = [count * planes]
```

10.71.4.20 void QueuedCUDATracker::GetRadialZLUTSize (int & count, int & planes, int & radialSteps) [override], [virtual]

Implements [QueuedTracker](#).

```
871 {
872     count = devices[0]->radial_zlut.count;
873     planes = devices[0]->radial_zlut.h;
874     rsteps = cfg.zlut_radialsteps;
875 }
```

10.71.4.21 QueuedCUDATracker::Stream * QueuedCUDATracker::GetReadyStream () [protected]

```
376 {
377     while (true) {
378         jobQueueMutex.lock();
379
380         Stream *best = 0;
381         for (uint i=0;i<streams.size();i++)
382         {
383             Stream*s = streams[i];
384             if (s->state == Stream::StreamIdle) {
385                 if (!best || (s->JobCount() > best->JobCount()))
386                     best = s;
387             }
388         }
389
390         jobQueueMutex.unlock();
391
392         if (best)
393             return best;
394
395         Threads::Sleep(1);
396     }
397 }
```

10.71.4.22 int QueuedCUDATracker::GetResultCount () [override], [virtual]

Implements [QueuedTracker](#).

```
878 {
879     resultMutex.lock();
880     int r = resultCount;
881     resultMutex.unlock();
882     return r;
883 }
```

10.71.4.23 void QueuedCUDATracker::InitializeDeviceList() [protected]

```

113 {
114     int numDevices;
115     cudaGetDeviceCount(&numDevices);
116
117     // Select the most powerful one
118     if (cfg.cuda_device == QTrkCUDA_UseBest) {
119         cfg.cuda_device = GetBestCUDAdevice();
120         devices.push_back(new Device(cfg.cuda_device));
121     } else if (cfg.cuda_device == QTrkCUDA_UseAll) {
122         // Use all devices
123         for (int i=0;i<numDevices;i++) {
124             devices.push_back(new Device(i));
125         }
126     } else if (cfg.cuda_device == QTrkCUDA_UseList) {
127         for (uint i=0;i<cudaDeviceList.size();i++) {
128             devices.push_back(new Device(cudaDeviceList[i]));
129         }
130     }
131     deviceReport = "Using devices: ";
132     for (uint i=0;i<devices.size();i++) {
133         cudaDeviceProp p;
134         cudaGetDeviceProperties(&p, devices[i]->index);
135         deviceReport += SPrintf("%s%s", p.name, i<devices.size()-1?", ":"\\n");
136     }
137 }
```

10.71.4.24 bool QueuedCUDATracker::IsIdle() [override], [virtual]

Implements [QueuedTracker](#).

```

400 {
401     int ql = GetQueueLength(0);
402     return ql == 0;
403 }
```

10.71.4.25 void QueuedCUDATracker::ScheduleLocalization(void * data, int pitch, QTRK_PixelDataType pdt, const LocalizationJob * jobInfo) [override], [virtual]

Add a job to the queue to be processed. A job entails running the required algorithms on a single region of interest.

Parameters

in	<i>data</i>	Pointer to the data. Type specified by [pdt].
in	<i>pitch</i>	Distance in bytes between two successive rows of pixels (e.g. address of (0,0) - address of (0,1)).
in	<i>pdt</i>	Type of [data], specified by QTRK_PixelDataType .
in	<i>jobInfo</i>	Structure with metadata for the ROI to be handled. See LocalizationJob .

Implements [QueuedTracker](#).

```

432 {
433     Stream* s = GetReadyStream();
434
435     jobQueueMutex.lock();
436     int jobIndex = s->jobs.size();
437     LocalizationJob job = *jobInfo;
438     s->jobs.push_back(job);
439     s->localizeFlags = localizeMode; // which kernels to run
440     s->locParams[jobIndex].zlutIndex = jobInfo->zlutIndex;
441
442     if (s->jobs.size() == batchSize)
```

```

443     s->state = Stream::StreamPendingExec;
444     jobQueueMutex.unlock();
445
446     s->imageBufMutex.lock();
447     // Copy the image to the batch image buffer (CPU side)
448     float* hostbuf = &s->hostImageBuf[cfg.height*cfg.width*jobIndex];
449     CopyImageToFloat( (uchar*)data, cfg.width, cfg.
450     height, pitch, pdt, hostbuf);
451     if (localizeMode & LT_ClearFirstFourPixels) {
452         for(int i=0;i<4;i++) hostbuf[i]=0;
453     }
454     s->imageBufMutex.unlock();
455     //dbgprintf("Job: %d\n", jobIndex);
456 }
```

10.71.4.26 void QueuedCUDATracker::SchedulingThreadEntryPoint (void * param) [static], [protected]

```

237 {
238     ((QueuedCUDATracker*)param)->SchedulingThreadMain();
239 }
```

10.71.4.27 void QueuedCUDATracker::SchedulingThreadMain () [protected]

```

242 {
243     std::vector<Stream*> activeStreams;
244
245     while (!quitScheduler) {
246         jobQueueMutex.lock();
247         Stream* s = 0;
248         for (uint i=0;i<streams.size();i++)
249             if (streams[i]->state == Stream::StreamPendingExec) {
250                 s=streams[i];
251                 s->state = Stream::StreamExecuting;
252                 // dbgprintf("Executing stream %p [%d]. %d jobs\n", s, i, s->JobCount());
253                 break;
254             }
255         jobQueueMutex.unlock();
256
257         if (s) {
258             s->imageBufMutex.lock();
259
260             // Launch filled batches, or if flushing launch every batch with nonzero jobs
261             if (useTextureCache)
262                 ExecuteBatch<ImageSampler_Tex> (s);
263             else
264                 ExecuteBatch<ImageSampler_MemCopy> (s);
265             s->imageBufMutex.unlock();
266             activeStreams.push_back(s);
267         }
268
269         // Fetch results
270         for (uint a=0;a<activeStreams.size();a++) {
271             Stream* s = activeStreams[a];
272             if (s->IsExecutionDone()) {
273                 // dbgprintf("Stream %p done.\n", s);
274                 CopyStreamResults(s);
275                 s->localizeFlags = 0; // reset this for the next batch
276                 jobQueueMutex.lock();
277                 s->jobs.clear();
278                 s->state = Stream::StreamIdle;
279                 jobQueueMutex.unlock();
280                 activeStreams.erase(std::find(activeStreams.begin(),activeStreams.end(),s));
281                 break;
282             }
283         }
284
285         Threads::Sleep(1);
286     }
287 }
```

10.71.4.28 void QueuedCUDATracker::SetConfigValue (std::string *name*, std::string *value*) [override], [virtual]

Implements [QueuedTracker](#).

```
913 {
914     if (name == "use_texturecache")
915         useTextureCache = atoi(value.c_str()) != 0;
916 }
```

10.71.4.29 void QueuedCUDATracker::SetLocalizationMode (LocMode_t *locType*) [override], [virtual]

Select which algorithm is to be used.

Parameters

in	<i>locType</i>	An integer used as a bitmask for settings based on LocalizeModeEnum .
----	----------------	---

Implements [QueuedTracker](#).

```
422 {
423     Flush();
424     while (!IsIdle());
425
426     jobQueueMutex.lock();
427     localizeMode = mode;
428     jobQueueMutex.unlock();
429 }
```

10.71.4.30 void QueuedCUDATracker::SetPixelCalibrationFactors (float *offsetFactor*, float *gainFactor*) [override], [virtual]

Implements [QueuedTracker](#).

```
778 {
779     gc_mutex.lock();
780     gc_gainFactor = gainFactor;
781     gc_offsetFactor = offsetFactor;
782     gc_mutex.unlock();
783 }
```

10.71.4.31 void QueuedCUDATracker::SetPixelCalibrationImages (float * *offset*, float * *gain*) [override], [virtual]

Implements [QueuedTracker](#).

```
743 {
744     for (uint i=0;i<devices.size();i++) {
745         devices[i]->SetPixelCalibrationImages(offset, gain, cfg.width,
746                                                 cfg.height);
747     }
748     // Copy to CPU side buffers for BuildLUT
749     int nelem = devices[0]->radial_zlut.count * cfg.width * cfg.
```

```

height;
750   if (offset && gc_offset.size()!=nelem) {
751     gc_offset.resize(nelem);
752     gc_offset.assign(offset,offset+nelem);
753   }
754   if (!offset) gc_offset.clear();
755
756   if (gain && gc_gain.size()!=nelem) {
757     gc_gain.reserve(nelem);
758     gc_gain.assign(gain, gain+nelem);
759   }
760   if (!gain) gc_gain.clear();
761 }
```

10.71.4.32 void QueuedCUDATracker::SetRadialWeights (float * zcmp) [override], [virtual]

Implements [QueuedTracker](#).

```

823 {
824   for (uint i=0;i<devices.size();i++) {
825     devices[i]->SetRadialWeights(zcmp);
826   }
827 }
```

10.71.4.33 void QueuedCUDATracker::SetRadialZLUT (float * data, int numLUTs, int planes) [override], [virtual]

Implements [QueuedTracker](#).

```

812 {
813   for (uint i=0;i<devices.size();i++) {
814     devices[i]->SetRadialZLUT(data, cfg.zlut_radialsteps, planes, numLUTs);
815   }
816
817   for (uint i=0;i<streams.size();i++) {
818     StreamUpdateZLUTSize(streams[i]);
819   }
820 }
```

10.71.4.34 void QueuedCUDATracker::StreamUpdateZLUTSize (Stream * s) [protected]

```

830 {
831   cudaSetDevice(s->device->index);
832   s->d_zlutzmpscores.init(s->device->radial_zlut.h * batchSize);
833   // radialsteps never changes
834 }
```

10.71.4.35 dim3 QueuedCUDATracker::threads () [inline], [protected]

```
190 { return dim3(numThreads); }
```

10.71.5 Member Data Documentation

10.71.5.1 int QueuedCUDATracker::batchesDone

10.71.5.2 int QueuedCUDATracker::batchSize [protected]

Amount of images to be sent at once per stream.

Higher batchsize = higher speeds. Reason why it's faster:

1. More threads & blocks are created, allowing more efficient memory latency hiding by warp switching, or in other words, higher occupancy is achieved.
2. Bigger batches are copied at a time, achieving higher effective PCIe bus bandwidth.

10.71.5.3 KernelProfileTime QueuedCUDATracker::cpu_time

10.71.5.4 cudaDeviceProp QueuedCUDATracker::deviceProp [protected]

10.71.5.5 std::string QueuedCUDATracker::deviceReport

10.71.5.6 std::vector<Device*> QueuedCUDATracker::devices [protected]

10.71.5.7 std::vector<float> QueuedCUDATracker::gc_gain [protected]

10.71.5.8 float QueuedCUDATracker::gc_gainFactor [protected]

10.71.5.9 Threads::Mutex QueuedCUDATracker::gc_mutex [protected]

10.71.5.10 std::vector<float> QueuedCUDATracker::gc_offset [protected]

10.71.5.11 float QueuedCUDATracker::gc_offsetFactor [protected]

10.71.5.12 Threads::Mutex QueuedCUDATracker::jobQueueMutex [protected]

10.71.5.13 LocMode_t QueuedCUDATracker::localizeMode [protected]

10.71.5.14 int QueuedCUDATracker::numThreads [protected]

10.71.5.15 QI QueuedCUDATracker::qalign [protected]

10.71.5.16 QI QueuedCUDATracker::qi [protected]

10.71.5.17 Atomic<bool> QueuedCUDATracker::quitScheduler [protected]

10.71.5.18 int QueuedCUDATracker::resultCount [protected]

10.71.5.19 `Threads::Mutex QueuedCUDATracker::resultMutex` [protected]

10.71.5.20 `std::list<LocalizationResult> QueuedCUDATracker::results` [protected]

10.71.5.21 `Threads::Handle* QueuedCUDATracker::schedulingThread` [protected]

10.71.5.22 `std::vector<Stream*> QueuedCUDATracker::streams` [protected]

10.71.5.23 `KernelProfileTime QueuedCUDATracker::time`

10.71.5.24 `bool QueuedCUDATracker::useTextureCache` [protected]

10.71.5.25 `uint QueuedCUDATracker::zlut_build_flags` [protected]

cpu size buffers are needed because BuildLUT uses CPU tracking

The documentation for this class was generated from the following files:

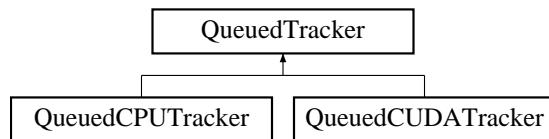
- cudatrack/[QueuedCUDATracker.h](#)
- cudatrack/[QueuedCUDATracker.cu](#)

10.72 QueuedTracker Class Reference

Abstract tracker interface, implemented by [QueuedCUDATracker](#) and [QueuedCPUTracker](#).

```
#include <QueuedTracker.h>
```

Inheritance diagram for QueuedTracker:



Public Types

- `typedef std::map< std::string, std::string > ConfigValueMap`

Public Member Functions

- `QueuedTracker ()`
- `virtual ~QueuedTracker ()`
- `virtual void SetLocalizationMode (LocMode_t locType)=0`
`Select which algorithm is to be used.`
- `virtual void SetPixelCalibrationImages (float *offset, float *gain)=0`
- `virtual void SetPixelCalibrationFactors (float offsetFactor, float gainFactor)=0`
- `virtual void ScheduleLocalization (void *data, int pitch, QTRK_PixelDataType pdt, const LocalizationJob *jobInfo)=0`
`Add a job to the queue to be processed. A job entails running the required algorithms on a single region of interest.`
- `void ScheduleImageData (ImageData *data, const LocalizationJob *jobInfo)`
`Quick function to schedule a single ROI from an ImageData object.`
- `virtual void ClearResults ()=0`
`Clear results.`
- `virtual void Flush ()=0`
`Stop waiting for more jobs to do, and just process the current batch.`
- `virtual int ScheduleFrame (void *imgptr, int pitch, int width, int height, ROIPosition *positions, int numROI, QTRK_PixelDataType pdt, const LocalizationJob *jobInfo)`
- `virtual void SetRadialZLUT (float *data, int count, int planes)=0`
- `virtual void GetRadialZLUT (float *dst)=0`
- `virtual void GetRadialZLUTSize (int &count, int &planes, int &radialsteps)=0`
- `virtual void SetRadialWeights (float *zcmp)=0`
- `virtual void EnableRadialZLUTCompareProfile (bool enabled)=0`
- `virtual void GetRadialZLUTCompareProfile (float *dst)=0`
- `virtual void GetImageZLUTSize (int *dims)`
- `virtual void GetImageZLUT (float *dst)`
- `virtual bool SetImageZLUT (float *dst, float *radial_zlut, int *dims)`
- `virtual void BeginLUT (uint flags)=0`
- `virtual void BuildLUT (void *data, int pitch, QTRK_PixelDataType pdt, int plane, vector2f *known_pos=0)=0`
- `virtual void FinalizeLUT ()=0`
- `virtual int GetResultCount ()=0`
- `virtual int FetchResults (LocalizationResult *results, int maxResults)=0`
- `virtual int GetQueueLength (int *maxQueueLen=0)=0`
- `virtual bool IsIdle ()=0`
- `virtual void SetConfigValue (std::string name, std::string value)=0`
- `virtual ConfigValueMap GetConfigValues ()=0`
- `virtual std::string GetProfileReport ()`
- `virtual std::string GetWarnings ()`
- `virtual bool GetDebugImage (int ID, int *w, int *h, float **pData)`
- `ImageData DebugImage (int ID)`
- `void ScheduleLocalization (uchar *data, int pitch, QTRK_PixelDataType pdt, uint frame, uint timestamp, vector3f *initial, uint zlutIndex)`
- `void ComputeZBiasCorrection (int bias_planes, CImageData *result, int smpPerPixel, bool useSplineInterp)`
- `float ZLUTBiasCorrection (float z, int zlut_planes, int bead)`
- `void SetZLUTBiasCorrection (const CImageData &data)`
- `CImageData * GetZLUTBiasCorrection ()`

Public Attributes

- `QTrkComputedConfig cfg`

Protected Attributes

- `CImageData * zlut_bias_correction`

10.72.1 Detailed Description

Abstract tracker interface, implemented by [QueuedCUDATracker](#) and [QueuedCPUTracker](#).

10.72.2 Member Typedef Documentation

10.72.2.1 `typedef std::map<std::string, std::string> QueuedTracker::ConfigValueMap`

10.72.3 Constructor & Destructor Documentation

10.72.3.1 `QueuedTracker::QueuedTracker()`

```
95 {  
96     zlut_bias_correction=0;  
97 }
```

10.72.3.2 `QueuedTracker::~QueuedTracker() [virtual]`

```
100 {  
101 }
```

10.72.4 Member Function Documentation

10.72.4.1 `virtual void QueuedTracker::BeginLUT(uint flags) [pure virtual]`

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

10.72.4.2 `virtual void QueuedTracker::BuildLUT(void * data, int pitch, QTRK_PixelDataType pdt, int plane, vector2f * known_pos = 0) [pure virtual]`

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

10.72.4.3 `virtual void QueuedTracker::ClearResults() [pure virtual]`

Clear results.

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

10.72.4.4 void QueuedTracker::ComputeZBiasCorrection (int *bias_planes*, CImageData * *result*, int *smpPerPixel*, bool *useSplineInterp*)

```

174 {
175     int count,zlut_planes,radialsteps;
176     GetRadialZLUTSize(count, zlut_planes, radialsteps);
177     float* zlut_data = new float[count*zlut_planes*radialsteps];
178     GetRadialZLUT(zlut_data);
179
180     std::vector<float> qi_rweights = ComputeRadialBinWindow(
181         cfg.qi_radialsteps);
182     std::vector<float> zlut_rweights = ComputeRadialBinWindow(
183         cfg.zlut_radialsteps);
184
185     if (zlut_bias_correction)
186         delete zlut_bias_correction;
187
188     zlut_bias_correction = new CImageData(bias_planes, count);
189
190     parallel_for(count*bias_planes, [&](int job) {
191 //for (int job=0;job<count*bias_planes;job++) {
192         int bead = job/bias_planes;
193         int plane = job%bias_planes;
194
195         float *zlut_ptr = &zlut_data[ bead * (zlut_planes*radialsteps) ];
196         CPUTracker trk (cfg.width,cfg.height);
197         ImageData zlut(zlut_ptr, radialsteps, zlut_planes);
198         trk.SetRadialZLUT(zlut.data, zlut.h, zlut.w, 1, cfg.zlut_minradius,
199         cfg.zlut_maxradius, false, false);
200         trk.SetRadialWeights(&zlut_rweights[0]);
201
202         vector3f pos(cfg.width/2,cfg.height/2, plane /(float) bias_planes *
203         zlut_planes );
204         ImageData img = ImageData::alloc(cfg.width,
205         cfg.height);
206         GenerateImageFromLUT(&img, &zlut, cfg.
207         zlut_minradius, cfg.zlut_maxradius, pos, useSplineInterp,smpPerPixel);
208
209         bool bhit;
210         trk.SetImageFloat(img.data);
211         vector2f com = trk.ComputeMeanAndCOM();
212         vector2f qi = trk.ComputeQI(com, 2, cfg.qi_radialsteps,
213         cfg.qi_angstepspq, cfg.qi_angstep_factor,
214         cfg.qi_minradius, cfg.qi_maxradius, bhit, &qi_rweights[0]);
215         float z = trk.ComputeZ(qi, cfg.zlut_angularsteps, 0);
216         zlut_bias_correction->at(plane, bead) = z - pos.z;
217
218         //trk.ComputeRadialProfile(
219         img.free();
220         if ((job%(count*bias_planes/10)) == 0)
221             dbgprintf("job=%d\n", job);
222     // }
223     });
224
225     if (result)
226         *result = *zlut_bias_correction;
227 }

```

10.72.4.5 ImageData QueuedTracker::DebugImage (int *ID*)

```

119 {
120     ImageData img;
121     GetDebugImage(ID, &img.w, &img.h, &img.data);
122     return img;
123 }

```

10.72.4.6 virtual void QueuedTracker::EnableRadialZLUTCompareProfile (bool *enabled*) [pure virtual]

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

```
10.72.4.7 virtual int QueuedTracker::FetchResults ( LocalizationResult * results, int maxResults ) [pure  
virtual]
```

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

```
10.72.4.8 virtual void QueuedTracker::FinalizeLUT( ) [pure virtual]
```

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

```
10.72.4.9 virtual void QueuedTracker::Flush( ) [pure virtual]
```

Stop waiting for more jobs to do, and just process the current batch.

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

```
10.72.4.10 virtual ConfigValueMap QueuedTracker::GetConfigValues( ) [pure virtual]
```

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

```
10.72.4.11 virtual bool QueuedTracker::GetDebugImage ( int ID, int * w, int * h, float ** pData ) [inline],  
[virtual]
```

Reimplemented in [QueuedCPUTracker](#).

```
140 { return false; } // deallocate result with delete[]
```

```
10.72.4.12 virtual void QueuedTracker::GetImageZLUT ( float * dst ) [inline], [virtual]
```

Reimplemented in [QueuedCPUTracker](#).

```
116 {}
```

```
10.72.4.13 virtual void QueuedTracker::GetImageZLUTSize ( int * dims ) [inline], [virtual]
```

Reimplemented in [QueuedCPUTracker](#).

```
115 {}
```

```
10.72.4.14 virtual std::string QueuedTracker::GetProfileReport( ) [inline], [virtual]
```

Reimplemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

```
137 { return ""; }
```

10.72.4.15 virtual int QueuedTracker::GetQueueLength (int * *maxQueueLen* = 0) [pure virtual]

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

10.72.4.16 virtual void QueuedTracker::GetRadialZLUT (float * *dst*) [pure virtual]

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

10.72.4.17 virtual void QueuedTracker::GetRadialZLUTCompareProfile (float * *dst*) [pure virtual]

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

10.72.4.18 virtual void QueuedTracker::GetRadialZLUTSize (int & *count*, int & *planes*, int & *radialsteps*) [pure virtual]

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

10.72.4.19 virtual int QueuedTracker::GetResultCount () [pure virtual]

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

10.72.4.20 virtual std::string QueuedTracker::GetWarnings () [inline], [virtual]

```
138 { return ""; }
```

10.72.4.21 CImageData * QueuedTracker::GetZLUTBiasCorrection ()

```
228 {
229     if (zlut_bias_correction) return new CImageData(*
230         zlut_bias_correction);
231     return 0;
231 }
```

10.72.4.22 virtual bool QueuedTracker::IsIdle () [pure virtual]

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

10.72.4.23 int QueuedTracker::ScheduleFrame (void * *imgptr*, int *pitch*, int *width*, int *height*, ROIPosition * *positions*, int *numROI*, QTRK_PixelDataType *pdt*, const LocalizationJob * *jobInfo*) [virtual]

```

126 {
127     uchar* img = (uchar*)imgptr;
128     int bpp = PDT_BytesPerPixel(pdt);
129     int count=0;
130     for (int i=0;i<numROI;i++){
131         ROIPosition& pos = positions[i];
132
133         if (pos.x < 0 || pos.y < 0 || pos.x + cfg.width > width || pos.
134         y + cfg.height > height) {
135             dbgprintf("Skipping ROI %d. Outside of image.\n", i);
136             continue;
137         }
138         uchar *roiptr = &img[pitch * pos.y + pos.x * bpp];
139         LocalizationJob job = *jobInfo;
140         job.zlutIndex = i + jobInfo->zlutIndex; // used as offset
141         ScheduleLocalization(roiptr, pitch, pdt, &job);
142         count++;
143     }
144     return count;
145 }
```

10.72.4.24 void QueuedTracker::ScheduleImageData (ImageData * *data*, const LocalizationJob * *jobInfo*)

Quick function to schedule a single ROI from an [ImageData](#) object.

```

104 {
105     ScheduleLocalization(data->data, data->pitch(),
106     QTrkFloat, job);
107 }
```

10.72.4.25 virtual void QueuedTracker::ScheduleLocalization (void * *data*, int *pitch*, QTRK_PixelDataType *pdt*, const LocalizationJob * *jobInfo*) [pure virtual]

Add a job to the queue to be processed. A job entails running the required algorithms on a single region of interest.

Parameters

in	<i>data</i>	Pointer to the data. Type specified by [<i>pdt</i>].
in	<i>pitch</i>	Distance in bytes between two successive rows of pixels (e.g. address of (0,0) - address of (0,1)).
in	<i>pdt</i>	Type of [<i>data</i>], specified by QTRK_PixelDataType .
in	<i>jobInfo</i>	Structure with metadata for the ROI to be handled. See LocalizationJob .

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

10.72.4.26 void QueuedTracker::ScheduleLocalization (uchar * *data*, int *pitch*, QTRK_PixelDataType *pdt*, uint *frame*, uint *timestamp*, vector3f * *initial*, uint *zlutIndex*)

```

109 {
110     LocalizationJob j;
111     j.frame= frame;
112     j.timestamp = timestamp;
113     if (initial) j.initialPos = *initial;
114     j.zlutIndex = zlutIndex;
115     ScheduleLocalization(data,pitch,pdt,&j);
116 }
```

10.72.4.27 virtual void QueuedTracker::SetConfigValue (std::string *name*, std::string *value*) [pure virtual]

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

10.72.4.28 virtual bool QueuedTracker::SetImageZLUT (float * *dst*, float * *radial_zlut*, int * *dims*) [inline], [virtual]

Reimplemented in [QueuedCPUTracker](#).

```
117 { return false; }
```

10.72.4.29 virtual void QueuedTracker::SetLocalizationMode (LocMode_t *locType*) [pure virtual]

Select which algorithm is to be used.

Parameters

in	<i>locType</i>	An integer used as a bitmask for settings based on LocalizeModeEnum .
----	----------------	---

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

10.72.4.30 virtual void QueuedTracker::SetPixelCalibrationFactors (float *offsetFactor*, float *gainFactor*) [pure virtual]

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

10.72.4.31 virtual void QueuedTracker::SetPixelCalibrationImages (float * *offset*, float * *gain*) [pure virtual]

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

10.72.4.32 virtual void QueuedTracker::SetRadialWeights (float * *zcmp*) [pure virtual]

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

10.72.4.33 virtual void QueuedTracker::SetRadialZLUT (float * *data*, int *count*, int *planes*) [pure virtual]

Implemented in [QueuedCUDATracker](#), and [QueuedCPUTracker](#).

10.72.4.34 void QueuedTracker::SetZLUTBiasCorrection (const CImageData & *data*)

```
222 {
223     if (zlut_bias_correction) delete zlut_bias_correction;
224     zlut_bias_correction = new CImageData(bc);
225 }
```

10.72.4.35 float QueuedTracker::ZLUTBiasCorrection (float z, int zlut_planes, int bead)

```

148 {
149     if (!zlut_bias_correction)
150         return z;
151
152     /*
153     bias = d(r,4);
154     measured_pos = d(r,1) + bias;
155     pos = measured_pos;
156     for k=1:2
157         guess_bias = interp1(d(r,1), bias, pos);
158         pos = measured_pos - guess_bias;
159     end
160     */
161
162     // we have to reverse the bias table: we know that true_z + bias(true_z) = measured_z, but we can only
163     get bias(measured_z)
164     // It seems that one can iterate towards the right position:
165
166     float pos = z;
167     for (int k=0;k<4;k++) {
168         float tblpos = pos / (float)zlut_planes * zlut_bias_correction->
169         w;
170         float bias = zlut_bias_correction->interpolate1D(bead, tblpos);
171         pos = z - bias;
172     }
173     return pos;
174 }
```

10.72.5 Member Data Documentation

10.72.5.1 QTrkComputedConfig QueuedTracker::cfg

10.72.5.2 ClImageData* QueuedTracker::zlut_bias_correction [protected]

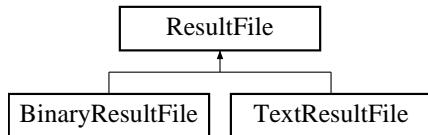
The documentation for this class was generated from the following files:

- cputrack/[QueuedTracker.h](#)
- cputrack/[QueuedTracker.cpp](#)

10.73 ResultFile Class Reference

```
#include <ResultManager.h>
```

Inheritance diagram for ResultFile:



Public Member Functions

- [ResultFile \(\)](#)
- virtual [~ResultFile \(\)](#)
- virtual void [LoadRow \(std::vector< vector3f > &pos\)=0](#)
- virtual void [SaveRow \(std::vector< vector3f > &pos\)=0](#)

10.73.1 Constructor & Destructor Documentation

10.73.1.1 `ResultFile::ResultFile() [inline]`

```
12 { }
```

10.73.1.2 `virtual ResultFile::~ResultFile() [inline], [virtual]`

```
13 {}
```

10.73.2 Member Function Documentation

10.73.2.1 `virtual void ResultFile::LoadRow(std::vector< vector3f > & pos) [pure virtual]`

Implemented in [BinaryResultFile](#), and [TextResultFile](#).

10.73.2.2 `virtual void ResultFile::SaveRow(std::vector< vector3f > & pos) [pure virtual]`

Implemented in [BinaryResultFile](#), and [TextResultFile](#).

The documentation for this class was generated from the following file:

- cputrack/[ResultManager.h](#)

10.74 ResultManager Class Reference

```
#include <ResultManager.h>
```

Classes

- struct [FrameCounters](#)
- struct [FrameResult](#)

Public Member Functions

- [ResultManager](#) (const char *outfile, const char *frameinfo, [ResultManagerConfig](#) *cfg, std::vector< std::string > colnames)
- [~ResultManager\(\)](#)
- void [SaveSection](#) (int start, int end, const char *beadposfile, const char *infofile)
- void [SetTracker](#) ([QueuedTracker](#) *qtrk)
- [QueuedTracker](#) * [GetTracker](#) ()
- int [GetBeadPositions](#) (int startFrame, int endFrame, int bead, [LocalizationResult](#) *r)
- int [GetResults](#) ([LocalizationResult](#) *results, int startFrame, int numResults)
- void [Flush](#) ()
- [FrameCounters](#) [GetFrameCounters](#) ()
- void [StoreFrameInfo](#) (int frame, double timestamp, float *columns)
- int [GetFrameCount](#) ()
- bool [RemoveBeadResults](#) (int bead)
- const [ResultManagerConfig](#) & [Config](#) ()

Protected Member Functions

- bool `CheckResultSpace` (int fr)
- void `Write` ()
- void `WriteBinaryResults` ()
- void `WriteTextResults` ()
- void `StoreResult` (LocalizationResult *r)
- bool `Update` ()
- void `WriteBinaryFileHeader` ()

Static Protected Member Functions

- static void `ThreadLoop` (void *param)

Protected Attributes

- Threads::Mutex `resultMutex`
- Threads::Mutex `trackerMutex`
- std::vector< std::string > `frameInfoNames`
- std::deque< FrameResult * > `frameResults`
- FrameCounters `cnt`
- ResultManagerConfig `config`
- ResultFile * `resultFile`
- QueuedTracker * `qtrk`
- std::string `outputFile`
- std::string `frameInfoFile`
- Threads::Handle * `thread`
- Atomic< bool > `quit`

10.74.1 Constructor & Destructor Documentation

10.74.1.1 ResultManager::ResultManager (const char * *outfile*, const char * *frameinfo*, ResultManagerConfig * *cfg*, std::vector< std::string > *colnames*)

```

43 {
44     config = *cfg;
45     outputFile = outfile;
46     this->frameInfoFile = frameInfoFile;
47
48     qtrk = 0;
49
50     thread = Threads::Create(ThreadLoop, this);
51     quit=false;
52
53     if (!outputFile.empty())
54         remove(outfile);
55     if (!this->frameInfoFile.empty())
56         remove(frameInfoFile);
57
58     frameInfoNames = colnames;
59
60     if (config.binaryOutput) {
61         WriteBinaryFileHeader();
62     }
63
64     dbgprintf("Allocating ResultManager with %d beads, %d motor columns and writeinterval %d\n",
65     cfg->numBeads, cfg->numFrameInfoColumns, cfg->
66     writeInterval);
67 }
```

10.74.1.2 ResultManager::~ResultManager()

```

96 {
97     quit = true;
98     Threads::WaitAndClose(thread);
99
100    DeleteAllElems(frameResults);
101 }

```

10.74.2 Member Function Documentation

10.74.2.1 bool ResultManager::CheckResultSpace(int fr) [protected]

```

362 {
363     if(fr < cnt.startFrame)
364         return false; // already removed, ignore
365
366     if (fr > cnt.processedFrames + 20000) {
367
368         dbgprintf("ResultManager: Ignoring suspiciously large frame number (%d).\n", fr);
369
370         return false;
371     }
372
373     while (fr >= cnt.startFrame + frameResults.size()) {
374
375         frameResults.push_back (new FrameResult( config.
376         numBeads, config.numFrameInfoColumns));
377     }
378
379     return true;
380 }

```

10.74.2.2 const ResultManagerConfig& ResultManager::Config() [inline]

```
85 { return config; }
```

10.74.2.3 void ResultManager::Flush()

```

308 {
309     trackerMutex.lock();
310     if (qtrk) qtrk->Flush();
311     trackerMutex.unlock();
312
313     Update();
314
315     resultMutex.lock();
316
317     Write();
318
319     // Dump stats about unfinished frames for debugging
320 #ifdef _DEBUG
321     for (uint i=0;i<frameResults.size();i++) {
322         FrameResult *fr = frameResults[i];
323 //         dbgprintf("Frame %d. TS: %f, Count: %d\n", i, fr->timestamp, fr->count);
324         if (fr->count != config.numBeads) {
325             for (int j=0;j<fr->results.size();j++) {
326                 if( fr->results[j].job.frame == 0)
327                     dbgprintf("%d, %d\n", i, j );
328             }
329             dbgprintf("\n");
330         }
331     }
332 #endif
333     resultMutex.unlock();
334 }

```

10.74.2.4 int ResultManager::GetBeadPositions (int startFrame, int endFrame, int bead, LocalizationResult * r)

```

286 {
287     int count = end-startfr;
288
289     resultMutex.lock();
290     if (end > cnt.processedFrames)
291         end = cnt.processedFrames;
292
293     int start = startfr - cnt.startFrame;
294     if (start < 0) start = 0;
295     if (count > cnt.processedFrames-cnt.startFrame)
296         count = cnt.processedFrames-cnt.startFrame;
297
298     for (int i=0;i<count;i++){
299         results[i] = frameResults[i+start]->results[bead];
300     }
301     resultMutex.unlock();
302
303     return count;
304 }
```

10.74.2.5 int ResultManager::GetFrameCount ()

```

400 {
401     resultMutex.lock();
402     int nfr = cnt.capturedFrames;
403     resultMutex.unlock();
404     return nfr;
405 }
```

10.74.2.6 ResultManager::FrameCounters ResultManager::GetFrameCounters ()

```

338 {
339     resultMutex.lock();
340     FrameCounters c = cnt;
341     resultMutex.unlock();
342     return c;
343 }
```

10.74.2.7 int ResultManager::GetResults (LocalizationResult * results, int startFrame, int numResults)

```

346 {
347     resultMutex.lock();
348
349     if (startFrame >= cnt.startFrame && numFrames+startFrame <= cnt.
350         processedFrames) {
351         for (int f=0;f<numFrames;f++) {
352             int index = f + startFrame - cnt.startFrame;
353             for (int j=0;j<config.numBeads;j++)
354                 results[config.numBeads*f+j] = frameResults[index]->results[j];
355         }
356     resultMutex.unlock();
357
358     return numFrames;
359 }
```

10.74.2.8 QueuedTracker * ResultManager::GetTracker ()

```

208 {
209     trackerMutex.lock();
210     QueuedTracker* trk = qtrk;
211     trackerMutex.unlock();
212     return trk;
213 }
```

10.74.2.9 bool ResultManager::RemoveBeadResults (int *bead*)

```

408 {
409     // TODO: We need to modify the saved data file
410
411     for (uint i=0;i<frameResults.size();i++) {
412         auto fr = frameResults[i];
413
414         fr->count--;
415         fr->results.erase(fr->results.begin() + bead);
416     }
417
418     return true;
419 }
```

10.74.2.10 void ResultManager::SaveSection (int *start*, int *end*, const char * *beadposfile*, const char * *infofile*)10.74.2.11 void ResultManager::SetTracker (QueuedTracker * *qtrk*)

```

216 {
217     trackerMutex.lock();
218     this->qtrk = qtrk;
219     trackerMutex.unlock();
220 }
```

10.74.2.12 void ResultManager::StoreFrameInfo (int *frame*, double *timestamp*, float * *columns*)

```

380 {
381     resultMutex.lock();
382
383     if (CheckResultSpace(frame)) {
384         FrameResult* fr = frameResults[frame - cnt.startFrame];
385
386         if (!fr->hasFrameInfo) {
387             fr->timestamp = timestamp;
388             for (int i=0;i<config.numFrameInfoColumns;i++)
389                 fr->frameInfo[i] = columns[i];
390             ++cnt.capturedFrames;
391             fr->hasFrameInfo = true;
392         }
393     }
394
395     resultMutex.unlock();
396 }
```

10.74.2.13 void ResultManager::StoreResult (LocalizationResult * *r*) [protected]

```

105 {
106     if (CheckResultSpace(r->job.frame)) {
107
108         LocalizationResult scaled = *r;
109         // Make roi-centered pos
110         scaled.pos = scaled.pos - vector3f( qtrk->cfg.width*0.5f,
111                                         qtrk->cfg.height*0.5f, 0 );
112         scaled.pos = ( scaled.pos + config.offset ) * config.
113         scaling;
114         FrameResult* fr = frameResults[r->job.frame - cnt.
115         startFrame];
116         fr->results[r->jlutIndex] = scaled;
117         fr->count++;
118
119         // Advance processedFrames, either because measurements have been completed or because frames have
120         // been lost
121         while (cnt.processedFrames - cnt.startFrame < (int)
122 frameResults.size() &&
123         (frameResults[cnt.processedFrames - cnt.
124         startFrame]->count == config.numBeads || cnt.
125         capturedFrames - cnt.processedFrames > 1000 ))
```

```

119         {
120             if (frameResults[cnt.processedFrames-
121             cnt.startFrame]->count < config.numBeads)
122                 cnt.lostFrames++;
123             cnt.processedFrames++;
124         }
125     cnt.localizationDone++;
126 } else
127     cnt.lostFrames++;
128 }
129 }
```

10.74.2.14 void ResultManager::ThreadLoop (void * param) [static], [protected]

```

273 {
274     ResultManager* rm = (ResultManager*)param;
275
276     while(true) {
277         if (!rm->Update())
278             Threads::Sleep(20);
279
280         if (rm->quit)
281             break;
282     }
283 }
```

10.74.2.15 bool ResultManager::Update () [protected]

```

223 {
224     trackerMutex.lock();
225
226     if (!qtrk) {
227         trackerMutex.unlock();
228         return 0;
229     }
230
231     const int NResultBuf = 40;
232     LocalizationResult resultbuf[NResultBuf];
233
234     int count = qtrk->FetchResults( resultbuf, NResultBuf );
235
236     resultMutex.lock();
237     for (int i=0;i<count;i++)
238         StoreResult(&resultbuf[i]);
239
240     trackerMutex.unlock();
241
242     if (cnt.processedFrames - cnt.lastSaveFrame >=
243         config.writeInterval) {
244         Write();
245     }
246
247     if (config.maxFramesInMemory>0 && frameResults.size () >
248         config.maxFramesInMemory) {
249
250         int del = frameResults.size()-config.maxFramesInMemory;
251
252         if (cnt.processedFrames < cnt.startFrame) {
253             // write away any results that might be in there, unfinished localizations will be zero.
254             int lost = cnt.startFrame-cnt.processedFrames;
255             cnt.processedFrames += lost;
256             cnt.lostFrames += lost;
257
258             Write();
259         }
260
261         //dbgprintf("Removing %d frames from memory\n", del);
262
263         for (int i=0;i<del;i++)
264             delete frameResults[i];
265         frameResults.erase(frameResults.begin(),
266                           frameResults.begin()+del);
267
268         cnt.startFrame += del;
269     }
270     resultMutex.unlock();
271
272     return count>0;
273 }
```

10.74.2.16 void ResultManager::Write() [protected]

```

194 {
195     resultMutex.lock();
196     if (config.binaryOutput)
197         WriteBinaryResults();
198     else
199         WriteTextResults();
200
201     dbgprintf("Saved frame %d to %d\n", cnt.lastSaveFrame,
202               cnt.processedFrames);
203     cnt.lastSaveFrame = cnt.processedFrames;
204
205     resultMutex.unlock();
206 }
```

10.74.2.17 void ResultManager::WriteBinaryFileHeader() [protected]

```

68 {
69     FILE* f = fopen(outputFile.c_str(), "wb");
70     if (!f) {
71         throw std::runtime_error( SPrintf("Unable to open file %s",
72                                         outputFile.c_str()));
73     }
74
75     // Write file header
76     int version = BINFILE_VERSION;
77     fwrite(&version, sizeof(int), 1, f);
78     fwrite(&config.numBeads, sizeof(int), 1, f);
79     fwrite(&config.numFrameInfoColumns, sizeof(int), 1, f);
80     long data_offset_pos = ftell(f);
81     int tmp=1234;
82     fwrite(&tmp,sizeof(int), 1,f);
83     for (int i=0;i<config.numFrameInfoColumns;i++) {
84         auto& n = frameInfoNames[i];
85         fwrite(n.c_str(), n.length()+1, 1, f);
86     }
87     long data_offset = ftell(f);
88     dbgprintf("frame data offset: %d\n", data_offset);
89     fseek(f, data_offset_pos, SEEK_SET);
90     fwrite(&data_offset, sizeof(long), 1, f);
91
92     dbgprintf("writing %d beads and %d frame-info columns into file %s\n",
93               config.numBeads, config.numFrameInfoColumns,
94               outputFile.c_str());
95     fclose(f);
96 }
```

10.74.2.18 void ResultManager::WriteBinaryResults() [protected]

```

132 {
133     if (outputFile.empty())
134         return;
135
136     FILE* f = fopen(outputFile.c_str(), "ab");
137     if (!f) {
138         dbgprintf("ResultManager::WriteBinaryResult() Unable to open file %s\n",
139                   outputFile.c_str());
140         cnt.fileError++;
141     }
142
143     for (int j=cnt.lastSaveFrame; j<cnt.processedFrames;j++)
144     {
145         auto fr = frameResults[j-cnt.startFrame];
146         if (f) {
147             fwrite(&j, sizeof(uint), 1, f);
148             fwrite(&fr->timestamp, sizeof(double), 1, f);
149             fwrite(&fr->frameInfo[0], sizeof(float), config.
150                   numFrameInfoColumns, f);
151             for (int i=0;i<config.numBeads;i++)
152             {
153                 LocalizationResult *r = &fr->results[i];
154                 fwrite(&r->pos, sizeof(vector3f), 1, f);
155             }
156     }
```

```

155         for (int i=0;i<config.numBeads;i++)
156             fwrite(&fr->results[i].error, sizeof(int), 1, f);
157             for (int i=0;i<config.numBeads;i++) {
158                 fwrite(&fr->results[i].imageMean, sizeof(float), 1, f);
159             }
160     }
161 }
162 fclose(f);
163 }
```

10.74.2.19 void ResultManager::WriteTextResults() [protected]

```

166 {
167     FILE* f = outputFile.empty () ? 0 : fopen(outputFile.c_str(), "a");
168     FILE* finfo = frameInfoFile.empty() ? 0 : fopen(frameInfoFile.c_str(), "a");
169
170     for (int k=cnt.lastSaveFrame; k<cnt.processedFrames;k++)
171     {
172         auto fr = frameResults[k-cnt.startFrame];
173         if (f) {
174             fprintf(f,"%d\t%f\t", k, fr->timestamp);
175             for (int i=0;i<config.numBeads;i++)
176             {
177                 LocalizationResult *r = &fr->results[i];
178                 fprintf(f, "%.7f\t%.7f\t%.7f\t", r->pos.x,r->pos.y,r->
179 pos.z);
180             }
181             fputs("\n", f);
182         }
183         if (finfo) {
184             fprintf(finfo,"%d\t%f\t", k, fr->timestamp);
185             for (int i=0;i<config.numFrameInfoColumns;i++)
186                 fprintf(finfo, "%.7f\t", fr->frameInfo[i]);
187             fputs("\n", finfo);
188         }
189         if(finfo) fclose(finfo);
190         if(f) fclose(f);
191 }
```

10.74.3 Member Data Documentation

10.74.3.1 FrameCounters **ResultManager::cnt** [protected]

10.74.3.2 ResultManagerConfig **ResultManager::config** [protected]

10.74.3.3 std::string **ResultManager::frameInfoFile** [protected]

10.74.3.4 std::vector<std::string> **ResultManager::frameInfoNames** [protected]

10.74.3.5 std::deque<FrameResult*> **ResultManager::frameResults** [protected]

10.74.3.6 std::string **ResultManager::outputFile** [protected]

10.74.3.7 QueuedTracker* **ResultManager::qtrk** [protected]

10.74.3.8 Atomic<bool> **ResultManager::quit** [protected]

10.74.3.9 ResultFile* **ResultManager::resultFile** [protected]

10.74.3.10 Threads::Mutex **ResultManager::resultMutex** [protected]

10.74.3.11 Threads::Handle* **ResultManager::thread** [protected]

10.74.3.12 Threads::Mutex **ResultManager::trackerMutex** [protected]

The documentation for this class was generated from the following files:

- cputrack/[ResultManager.h](#)
- cputrack/[ResultManager.cpp](#)

10.75 ResultManagerConfig Struct Reference

```
#include <ResultManager.h>
```

Public Attributes

- int [numBeads](#)
- int [numFrameInfoColumns](#)
- [vector3f](#) [scaling](#)
- [vector3f](#) [offset](#)
- int [writeInterval](#)
- uint [maxFramesInMemory](#)
- uint8_t [binaryOutput](#)

10.75.1 Member Data Documentation

10.75.1.1 uint8_t [ResultManagerConfig::binaryOutput](#)

10.75.1.2 uint [ResultManagerConfig::maxFramesInMemory](#)

10.75.1.3 int [ResultManagerConfig::numBeads](#)

10.75.1.4 int [ResultManagerConfig::numFrameInfoColumns](#)

10.75.1.5 [vector3f](#) [ResultManagerConfig::offset](#)

10.75.1.6 [vector3f](#) [ResultManagerConfig::scaling](#)

10.75.1.7 int [ResultManagerConfig::writeInterval](#)

The documentation for this struct was generated from the following file:

- cputrack/[ResultManager.h](#)

10.76 ROIPosition Struct Reference

Struct used to define the top-left corner position of an ROI within a frame. ROI is [x .. x+w ; y .. y+h].

```
#include <qtrk_c_api.h>
```

Public Attributes

- int `x`
- int `y`

10.76.1 Detailed Description

Struct used to define the top-left corner position of an ROI within a frame. ROI is [`x .. x+w ; y .. y+h`].

10.76.2 Member Data Documentation

10.76.2.1 int ROIPosition::x

10.76.2.2 int ROIPosition::y

The documentation for this struct was generated from the following file:

- cputrack/qtrk_c_api.h

10.77 RunTrackerResults Struct Reference

```
#include <SharedTests.h>
```

Public Member Functions

- void `computeStats ()`

Public Attributes

- std::vector< `vector3f` > `output`
- std::vector< `vector3f` > `truepos`
- `vector3f meanErr`
- `vector3f stdev`

10.77.1 Member Function Documentation

10.77.1.1 void RunTrackerResults::computeStats () [inline]

```
338
339     MeanStDevError(truepos, output, meanErr,
340     stdev);
340 }
```

10.77.2 Member Data Documentation

10.77.2.1 `vector3f RunTrackerResults::meanErr`

10.77.2.2 `std::vector<vector3f> RunTrackerResults::output`

10.77.2.3 `vector3f RunTrackerResults::stdev`

10.77.2.4 `std::vector<vector3f> RunTrackerResults::truepos`

The documentation for this struct was generated from the following file:

- `cputrack-test/SharedTests.h`

10.78 SampleFisherMatrix Class Reference

```
#include <FisherMatrix.h>
```

Public Member Functions

- `SampleFisherMatrix (float maxValue)`
- `Matrix3X3 Compute (vector3f pos, vector3f delta, ImageData &lut, int w, int h, float lutMinRadius, float lutMaxRadius)`
- `Matrix3X3 Compute (vector3f pos, vector3f delta, int w, int h, std::function< void(ImageData &out, vector3f pos)> imageGenerator)`
- `Matrix3X3 ComputeAverageFisher (vector3f pos, int Nsamples, vector3f sampleRange, vector3f delta, int w, int h, std::function< void(ImageData &out, vector3f pos)> imggen)`

Private Member Functions

- `template<typename T > void ImgDeriv (TImageData< T > &imgpx, TImageData< T > &imgmx, T delta)`
- `template<typename T > double FisherElem (TImageData< T > &mu, TImageData< T > &muderiv1, TImageData< T > &muderiv2)`

Private Attributes

- `float maxValue`

10.78.1 Constructor & Destructor Documentation

10.78.1.1 `SampleFisherMatrix::SampleFisherMatrix (float maxValue) [inline]`

```
34      {
35          this->maxValue = maxValue;
36      }
```

10.78.2 Member Function Documentation

10.78.2.1 Matrix3X3 SampleFisherMatrix::Compute (**vector3f pos, vector3f delta, ImageData & lut, int w, int h, float zlutMinRadius, float zlutMaxRadius**) [inline]

```

39      {
40          return Compute(pos, delta, w,h,[&] (ImageData& out,
41          vector3f pos) {
42              ImageData tmp = ImageData::alloc(w,h);
43              GenerateImageFromLUT(&tmp, &lut, zlutMinRadius, zlutMaxRadius, pos);
44              out.set(tmp);
45              tmp.free();
46      });
47  }

```

10.78.2.2 Matrix3X3 SampleFisherMatrix::Compute (**vector3f pos, vector3f delta, int w, int h, std::function<void(ImageData &out, vector3f pos)> imageGenerator**) [inline]

```

49      {
50          static int t=0;
51
52          // compute derivatives
53          auto mu = ImageData::alloc(w,h);
54          imageGenerator(mu, pos);
55
56          float maxImg = mu.mean();
57
58          auto imgpx = ImageData::alloc(w,h);
59          auto imgpy = ImageData::alloc(w,h);
60          auto imgpz = ImageData::alloc(w,h);
61          imageGenerator(imgpx, vector3f(pos.x+delta.x, pos.y,pos.z));
62          imageGenerator(imgpy, vector3f(pos.x, pos.y+delta.y,pos.z));
63          imageGenerator(imgpz, vector3f(pos.x, pos.y,pos.z+delta.z));
64
65 //          WriteImageAsCSV("imgpx.csv", imgpx.data, imgpx.w, imgpx.h);
66 //          WriteImageAsCSV("imgpy.csv", imgpy.data, imgpx.w, imgpx.h);
67 //          WriteImageAsCSV("imgpz.csv", imgpz.data, imgpx.w, imgpx.h);
68
69          auto imgmx = ImageData::alloc(w,h);
70          auto imgmy = ImageData::alloc(w,h);
71          auto imgmz = ImageData::alloc(w,h);
72          imageGenerator(imgmx, vector3f(pos.x-delta.x, pos.y,pos.z));
73          imageGenerator(imgmy, vector3f(pos.x, pos.y-delta.y,pos.z));
74          imageGenerator(imgmz, vector3f(pos.x, pos.y,pos.z-delta.z));
75          //WriteImageAsCSV("imgmx.csv", imgmx.data, imgpx.w, imgpx.h);
76          //WriteImageAsCSV("imgmy.csv", imgmy.data, imgpx.w, imgpx.h);
77          //WriteImageAsCSV("imgmz.csv", imgmz.data, imgpx.w, imgpx.h);
78
79          ImgDeriv(imgpx, imgmx, delta.x);
80          ImgDeriv(imgpy, imgmy, delta.y);
81          ImgDeriv(imgpz, imgmz, delta.z);
82
83 //          WriteJPEGFile("mu_dx.jpg", imgpx);
84 //          WriteJPEGFile("mu_dy.jpg", imgpy);
85 //          WriteJPEGFile("mu_dz.jpg", imgpz);
86          t++;
87
88          double Ixx = FisherElem(mu, imgpx, imgpx);
89          double Iyy = FisherElem(mu, imgpy, imgpy);
90          double Izz = FisherElem(mu, imgpz, imgpz);
91          double Ixy = FisherElem(mu, imgpx, imgpy);
92          double Ixz = FisherElem(mu, imgpx, imgpz);
93          double Iyz = FisherElem(mu, imgpy, imgpz);
94
95          Matrix3X3 fisher;
96          fisher(0,0) = Ixx; fisher(0,1) = Ixy; fisher(0,2) = Ixz;
97          fisher(1,0) = Ixy; fisher(1,1) = Iyy; fisher(1,2) = Iyz;
98          fisher(2,0) = Ixz; fisher(2,1) = Iyz; fisher(2,2) = Izz;
99          fisher *= maxValue / maxImg;
100
101          imgpx.free(); imgpy.free(); imgpz.free();
102          imgmx.free(); imgmy.free(); imgmz.free();
103          mu.free();
104          return fisher;
105      }

```

10.78.2.3 Matrix3X3 SampleFisherMatrix::ComputeAverageFisher (`vector3f pos, int Nsamples, vector3f sampleRange, vector3f delta, int w, int h, std::function< void(ImageData &out, vector3f pos) > imggen`) [inline]

```

108     {
109         Matrix3X3* results = new Matrix3X3[Nsamples];
110
111         auto f = [&] (int index) {
112             vector3f smpPos = pos + sampleRange*vector3f(rand_uniform<float>() - 0.5f,
113             rand_uniform<float>() - 0.5f, rand_uniform<float>());
114             //vector3f smpPos = pos + ( vector3f(rand_normal<float>(), rand_normal<float>(),
115             rand_normal<float>()) * initialStDev);
116             results[index] = Compute(smpPos, delta, w,h, imggen);
117             //    dbgprintf("%d done.\n", index);
118         };
119
120         if (0) {
121             ThreadPool<int, std::function<void (int index)>
122             > pool(f);
123
124             for (int i=0;i<Nsamples;i++)
125                 pool.AddWork(i);
126             pool.WaitUntilDone();
127         } else {
128             for (int i=0;i<Nsamples;i++)
129                 f(i);
130
131             Matrix3X3 accum;
132             for (int i=0;i<Nsamples;i++)
133                 accum += results[i];
134             delete[] results;
135
136             Matrix3X3 matrix = accum;
137             matrix *= 1.0f/Nsamples;
138             return matrix;
139         }
140     }

```

10.78.2.4 template<typename T > double SampleFisherMatrix::FisherElem (`TImageData< T > & mu, TImageData< T > & muderiv1, TImageData< T > & muderiv2`) [inline], [private]

```

23     {
24         double sum = 0;
25         for (int y=0;y<mu.h;y++) {
26             for (int x=0;x<mu.w;x++)
27                 sum += (double)muderiv1.at(x,y) * (double) muderiv2.at(x,y) / (1e-5f + (double)mu.
28                 at(x,y));
29         }
30         return sum;
31     }

```

10.78.2.5 template<typename T > void SampleFisherMatrix::ImgDeriv (`TImageData< T > & imgpx, TImageData< T > & imgmx, T delta`) [inline], [private]

```

15     {
16         T inv2d = 1.0f/(2*delta);
17         for (int y=0;y<imgpx.w*imgpx.h;y++)
18             imgpx[y] = (imgpx[y] - imgmx[y]) * inv2d;
19     }

```

10.78.3 Member Data Documentation

10.78.3.1 float SampleFisherMatrix::maxValue [private]

The documentation for this class was generated from the following file:

- cputrack/[FisherMatrix.h](#)

10.79 ScopedCPUProfiler Class Reference

Public Member Functions

- `ScopedCPUProfiler (double *time)`
- `~ScopedCPUProfiler ()`

Private Attributes

- `double * time`
- `double start`

10.79.1 Constructor & Destructor Documentation

10.79.1.1 `ScopedCPUProfiler::ScopedCPUProfiler (double * time) [inline]`

```
67             :  time(time)  {
68     start = GetPreciseTime();
69 }
```

10.79.1.2 `ScopedCPUProfiler::~ScopedCPUProfiler () [inline]`

```
70             {
71     double end = GetPreciseTime();
72     *time += start-end;
73 }
```

10.79.2 Member Data Documentation

10.79.2.1 `double ScopedCPUProfiler::start [private]`

10.79.2.2 `double* ScopedCPUProfiler::time [private]`

The documentation for this class was generated from the following file:

- `cudatrack/QueuedCUDATracker.cu`

10.80 SpeedAccResult Struct Reference

```
#include <SharedTests.h>
```

Public Member Functions

- `void Compute (const std::vector< vector3f > &results, std::function< vector3f(int x) > truepos)`

Public Attributes

- `vector3f acc`
- `vector3f bias`
- `vector3f crlb`
- `int speed`

10.80.1 Member Function Documentation

10.80.1.1 void SpeedAccResult::Compute (const std::vector< vector3f > & results, std::function< vector3f(int x) > truepos) [inline]

```

473
474
475     vector3f s;
476     for(uint i=0;i<results.size();i++) {
477         s+=results[i]-truepos(i);
478     }
479     s*=1.0f/results.size();
480     bias=s;
481
482     acc=vector3f();
483     for (uint i=0;i<results.size();i++) {
484         vector3f d = results[i]-truepos(i);
485         vector3f errMinusMeanErr = d - s;
486         acc += errMinusMeanErr*errMinusMeanErr;
487     }
488     acc = sqrt(acc/results.size());
489
490 }
```

10.80.2 Member Data Documentation

10.80.2.1 vector3f SpeedAccResult::acc

10.80.2.2 vector3f SpeedAccResult::bias

10.80.2.3 vector3f SpeedAccResult::crlb

10.80.2.4 int SpeedAccResult::speed

The documentation for this struct was generated from the following file:

- `cputrack-test/SharedTests.h`

10.81 SpeedInfo Struct Reference

Public Attributes

- `float speed_cpu`
- `float speed_gpu`
- `float sched_cpu`
- `float sched_gpu`

10.81.1 Member Data Documentation

10.81.1.1 float SpeedInfo::sched_cpu

10.81.1.2 float SpeedInfo::sched_gpu

10.81.1.3 float SpeedInfo::speed_cpu

10.81.1.4 float SpeedInfo::speed_gpu

The documentation for this struct was generated from the following file:

- cudatrack-test/test.cu

10.82 QueuedCUDATracker::Stream Struct Reference

```
#include <QueuedCUDATracker.h>
```

Public Types

- enum State { StreamIdle, StreamPendingExec, StreamExecuting }

Public Member Functions

- Stream (int streamIndex)
- ~Stream ()
- bool IsExecutionDone ()
- void OutputMemoryUse ()
- int JobCount ()

Public Attributes

- pinned_array< float3 > results
- pinned_array< float3 > com
- pinned_array< float > imgMeans
- pinned_array< LocalizationParams > locParams
- device_vec< LocalizationParams > d_locParams
- std::vector< LocalizationJob > jobs
- cudalImageListf images
- pinned_array< float > hostImageBuf
- Threads::Mutex imageBufMutex
- cudaStream_t stream
- cudaEvent_t localizationDone
- cudaEvent_t imageCopyDone
- cudaEvent_t comDone
- cudaEvent_t qjDone
- cudaEvent_t qalignDone

- cudaEvent_t `zcomputeDone`
- cudaEvent_t `imapDone`
- cudaEvent_t `batchStart`
- `device_vec< float3 > d_resultpos`
- `device_vec< float3 > d_com`
- `Ql::StreamInstance qi_instance`
- `Ql::StreamInstance qalign_instance`
- `device_vec< float > d_imgmeans`
- `device_vec< float > d_radialprofiles`
- `device_vec< float > d_zlutcmpscores`
- `uint localizeFlags`
- `Device * device`
- `State state`

10.82.1 Member Enumeration Documentation

10.82.1.1 enum QueuedCUDATracker::Stream::State

Enumerator

StreamIdle

StreamPendingExec

StreamExecuting

```
168         {
169         StreamIdle,
170         StreamPendingExec,
171         StreamExecuting
172     };
```

10.82.2 Constructor & Destructor Documentation

10.82.2.1 QueuedCUDATracker::Stream (int `streamIndex`)

```
290     : imageBufMutex(SPrintf("imagebuf%d", streamIndex).c_str())
291 {
292     device = 0;
293     hostImageBuf = 0;
294     images.data=0;
295     stream=0;
296     state=StreamIdle;
297     localizeFlags=0;
298 }
```

10.82.2.2 QueuedCUDATracker::Stream::~Stream ()

```
301 {
302     cudaSetDevice(device->index);
303
304     if(images.data) images.free();
305     cudaEventDestroy(localizationDone);
306     cudaEventDestroy(qiDone);
307     cudaEventDestroy(comDone);
308     cudaEventDestroy(imageCopyDone);
309     cudaEventDestroy(zcomputeDone);
310     cudaEventDestroy(imapDone);
311     cudaEventDestroy(batchStart);
312
313     if (stream)
314         cudaStreamDestroy(stream); // stream can be zero if in debugStream mode.
315 }
```

10.82.3 Member Function Documentation

10.82.3.1 bool QueuedCUDATracker::Stream::IsExecutionDone()

```
318 {
319     cudaSetDevice(device->index);
320     return cudaEventQuery(localizationDone) == cudaSuccess;
321 }
```

10.82.3.2 int QueuedCUDATracker::Stream::JobCount() [inline]

```
132 { return jobs.size(); }
```

10.82.3.3 void QueuedCUDATracker::Stream::OutputMemoryUse()

```
324 {
325     int deviceMem = d_com.memsize() + d_locParams.memsize() +
qi_instance.memsize() + d_radialprofiles.
memsize() +
326     d_resultpos.memsize() + d_zlutcmpscores.
memsize() + images.totalNumBytes();
327
328     int hostMem = hostImageBuf.memsize() + com.memsize() +
locParams.memsize() + results.memsize();
329
330     dbgprintf("Stream memory use: %d MB on host, %d MB device memory (%d for images). \n",
hostMem /1024/1024, deviceMem/1024/1024, images.totalNumBytes()/1024/1024);
331 }
```

10.82.4 Member Data Documentation

10.82.4.1 cudaEvent_t QueuedCUDATracker::Stream::batchStart

10.82.4.2 pinned_array<float3> QueuedCUDATracker::Stream::com

10.82.4.3 cudaEvent_t QueuedCUDATracker::Stream::comDone

10.82.4.4 device_vec<float3> QueuedCUDATracker::Stream::d_com

10.82.4.5 device_vec<float> QueuedCUDATracker::Stream::d_imgmeans

10.82.4.6 device_vec<LocalizationParams> QueuedCUDATracker::Stream::d_locParams

10.82.4.7 device_vec<float> QueuedCUDATracker::Stream::d_radialprofiles

10.82.4.8 device_vec<float3> QueuedCUDATracker::Stream::d_resultpos

10.82.4.9 device_vec<float> QueuedCUDATracker::Stream::d_zlutcmpscores

10.82.4.10 Device* QueuedCUDATracker::Stream::device

10.82.4.11 `pinned_array<float> QueuedCUDATracker::Stream::hostImageBuf`

10.82.4.12 `Threads::Mutex QueuedCUDATracker::Stream::imageBufMutex`

10.82.4.13 `cudaEvent_t QueuedCUDATracker::Stream::imageCopyDone`

10.82.4.14 `cudaImageListf QueuedCUDATracker::Stream::images`

10.82.4.15 `cudaEvent_t QueuedCUDATracker::Stream::imapDone`

10.82.4.16 `pinned_array<float> QueuedCUDATracker::Stream::imgMeans`

10.82.4.17 `std::vector<LocalizationJob> QueuedCUDATracker::Stream::jobs`

10.82.4.18 `cudaEvent_t QueuedCUDATracker::Stream::localizationDone`

10.82.4.19 `uint QueuedCUDATracker::Stream::localizeFlags`

10.82.4.20 `pinned_array<LocalizationParams> QueuedCUDATracker::Stream::locParams`

10.82.4.21 `QI::StreamInstance QueuedCUDATracker::Stream::qalign_instance`

10.82.4.22 `cudaEvent_t QueuedCUDATracker::Stream::qalignDone`

10.82.4.23 `QI::StreamInstance QueuedCUDATracker::Stream::qi_instance`

10.82.4.24 `cudaEvent_t QueuedCUDATracker::Stream::qiDone`

10.82.4.25 `pinned_array<float3> QueuedCUDATracker::Stream::results`

10.82.4.26 `State QueuedCUDATracker::Stream::state`

10.82.4.27 `cudaStream_t QueuedCUDATracker::Stream::stream`

10.82.4.28 `cudaEvent_t QueuedCUDATracker::Stream::zcomputeDone`

The documentation for this struct was generated from the following files:

- cudatrack/[QueuedCUDATracker.h](#)
- cudatrack/[QueuedCUDATracker.cu](#)

10.83 QI::StreamInstance Struct Reference

```
#include <QI.h>
```

Public Member Functions

- `~StreamInstance ()`
- `int memsize ()`

Public Attributes

- `device_vec< float > d_shiftbuffer`
- `device_vec< float2 > d_Qlprofiles`
- `device_vec< float2 > d_Qlprofiles_reverse`
- `device_vec< float > d_quadrants`
- `cufftHandle fftPlan`
- `cudaStream_t stream`

10.83.1 Constructor & Destructor Documentation

10.83.1.1 QI::StreamInstance::~StreamInstance () [inline]

```
36             {
37         cufftDestroy(fftPlan);
38     }
```

10.83.2 Member Function Documentation

10.83.2.1 int QI::StreamInstance::memsize () [inline]

```
40             {
41         size_t fftSize;
42         cufftGetSize(fftPlan, &fftSize);
43         return d_Qlprofiles.memsize() +
44             d_Qlprofiles_reverse.memsize() + d_quadrants.
45             memsize() + fftSize;
46     }
```

10.83.3 Member Data Documentation

10.83.3.1 device_vec<float2> QI::StreamInstance::d_Qlprofiles

10.83.3.2 device_vec<float2> QI::StreamInstance::d_Qlprofiles_reverse

10.83.3.3 device_vec<float> QI::StreamInstance::d_quadrants

10.83.3.4 device_vec<float> QI::StreamInstance::d_shiftbuffer

10.83.3.5 cufftHandle QI::StreamInstance::fftPlan

10.83.3.6 cudaStream_t QI::StreamInstance::stream

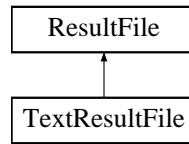
The documentation for this struct was generated from the following file:

- `cudatrack/QI.h`

10.84 TextResultFile Class Reference

```
#include <ResultManager.h>
```

Inheritance diagram for TextResultFile:



Public Member Functions

- [TextResultFile](#) (const char *fn, bool write)
- void [LoadRow](#) (std::vector< [vector3f](#) > &pos)
- void [SaveRow](#) (std::vector< [vector3f](#) > &pos)

Private Attributes

- FILE * f

10.84.1 Constructor & Destructor Documentation

10.84.1.1 [TextResultFile::TextResultFile \(const char * fn, bool write \)](#)

```
8 {
9     f = fopen(fn, write?"w":"r");
10 }
```

10.84.2 Member Function Documentation

10.84.2.1 [void TextResultFile::LoadRow \(std::vector< \[vector3f\]\(#\) > & pos \) \[virtual\]](#)

Implements [ResultFile](#).

```
13 {
14
15 }
```

10.84.2.2 [void TextResultFile::SaveRow \(std::vector< \[vector3f\]\(#\) > & pos \) \[virtual\]](#)

Implements [ResultFile](#).

```
18 {
19 }
```

10.84.3 Member Data Documentation

10.84.3.1 FILE* TextResultFile::f [private]

The documentation for this class was generated from the following files:

- cputrack/[ResultManager.h](#)
- cputrack/[ResultManager.cpp](#)

10.85 QueuedCPUTracker::Thread Struct Reference

Public Member Functions

- [Thread \(\)](#)
- void [lock \(\)](#)
- void [unlock \(\)](#)

Public Attributes

- [CPUTracker * tracker](#)
- [Threads::Handle * thread](#)
- [QueuedCPUTracker * manager](#)
- [Threads::Mutex * mutex](#)

10.85.1 Constructor & Destructor Documentation

10.85.1.1 QueuedCPUTracker::Thread::Thread() [inline]

```
61 { tracker=0; manager=0; thread=0; mutex=0; }
```

10.85.2 Member Function Documentation

10.85.2.1 void QueuedCPUTracker::Thread::lock() [inline]

```
67 { mutex->lock(); }
```

10.85.2.2 void QueuedCPUTracker::Thread::unlock() [inline]

```
68 { mutex->unlock(); }
```

10.85.3 Member Data Documentation

10.85.3.1 `QueuedCPUTracker* QueuedCPUTracker::Thread::manager`

10.85.3.2 `Threads::Mutex* QueuedCPUTracker::Thread::mutex`

10.85.3.3 `Threads::Handle* QueuedCPUTracker::Thread::thread`

10.85.3.4 `CPUTracker* QueuedCPUTracker::Thread::tracker`

The documentation for this struct was generated from the following file:

- cputrack/[QueuedCPUTracker.h](#)

10.86 ThreadPool< TWorkItem, TFunctor > Class Template Reference

```
#include <threads.h>
```

Public Member Functions

- [ThreadPool](#) (TFunctor f, int Nthreads=-1)
- [~ThreadPool](#) ()
- void [ProcessArray](#) (TWorkItem *items, int n)
- void [AddWork](#) (TWorkItem w)
- void [WaitUntilDone](#) ()
- bool [IsDone](#) ()
- void [Quit](#) ()

Protected Member Functions

- void [ItemDone](#) ()
- bool [GetNewItem](#) (TWorkItem &item)

Static Protected Member Functions

- static void [ThreadEntryPoint](#) (void *param)

Protected Attributes

- std::vector< [Threads::Handle](#) * > [threads](#)
- [Threads::Mutex](#) [workMutex](#)
- std::list< TWorkItem > [work](#)
- int [inProgress](#)
- Atomic< bool > [quit](#)
- TFunctor [worker](#)

10.86.1 Constructor & Destructor Documentation

10.86.1.1 template<typename TWorkItem, typename TFunctor> ThreadPool< TWorkItem, TFunctor >::ThreadPool (TFunctor *f*, int *Nthreads* = -1) [inline]

```

179                               : worker(f) {
180     if (Nthreads<0)
181       Nthreads = Threads::GetCPUCount();
182     threads.resize(Nthreads);
183     quit=false;
184     inProgress=0;
185     for (int i=0;i<Nthreads;i++)
186       threads[i]=Threads::Create(&ThreadEntryPoint,this);
187   }

```

10.86.1.2 template<typename TWorkItem, typename TFunctor> ThreadPool< TWorkItem, TFunctor >::~ThreadPool () [inline]

```

188   {
189     Quit();
190   }

```

10.86.2 Member Function Documentation

10.86.2.1 template<typename TWorkItem, typename TFunctor> void ThreadPool< TWorkItem, TFunctor >::AddWork (TWorkItem *w*) [inline]

```

195   {
196     workMutex.lock();
197     work.push_back(w);
198     workMutex.unlock();
199   }

```

10.86.2.2 template<typename TWorkItem, typename TFunctor> bool ThreadPool< TWorkItem, TFunctor >::GetNewItem (TWorkItem & *item*) [inline], [protected]

```

231   {
232     workMutex.lock();
233     bool r = !work.empty();
234     if (r) {
235       item = work.front();
236       work.pop_front();
237       inProgress++;
238     }
239     workMutex.unlock();
240     return r;
241   }

```

10.86.2.3 template<typename TWorkItem, typename TFunctor> bool ThreadPool< TWorkItem, TFunctor >::IsDone () [inline]

```

203   {
204     workMutex.lock();
205     bool r=work.empty() && inProgress==0;
206     workMutex.unlock();
207     return r;
208   }

```

10.86.2.4 template<typename TWorkItem, typename TFunctor> void ThreadPool< TWorkItem, TFunctor >::ItemDone() [inline], [protected]

```
226         {
227             workMutex.lock();
228             inProgress--;
229             workMutex.unlock();
230         }
```

10.86.2.5 template<typename TWorkItem, typename TFunctor> void ThreadPool< TWorkItem, TFunctor >::ProcessArray(TWorkItem * items, int n) [inline]

```
191         {
192             for(int i=0;i<n;i++)
193                 AddWork(items[i]);
194         }
```

10.86.2.6 template<typename TWorkItem, typename TFunctor> void ThreadPool< TWorkItem, TFunctor >::Quit() [inline]

```
209         {
210             quit=true;
211             for(uint i=0;i<threads.size();i++)
212                 Threads::WaitAndClose(threads[i]);
213             threads.clear();
214         }
```

10.86.2.7 template<typename TWorkItem, typename TFunctor> static void ThreadPool< TWorkItem, TFunctor >::ThreadEntryPoint(void * param) [inline], [static], [protected]

```
216             {
217                 ThreadPool* pool = ( ThreadPool * )param;
218                 TWorkItem item;
219                 while ( !pool->quit ) {
220                     if ( pool->GetNewItem(item) ) {
221                         pool->worker(item);
222                         pool->ItemDone();
223                     } else Threads::Sleep(1);
224                 }
225             }
```

10.86.2.8 template<typename TWorkItem, typename TFunctor> void ThreadPool< TWorkItem, TFunctor >::WaitUntilDone() [inline]

```
200             {
201                 while ( !IsDone() ) Threads::Sleep(1);
202             }
```

10.86.3 Member Data Documentation

- 10.86.3.1 template<typename TWorkItem, typename TFunctor> int **ThreadPool< TWorkItem, TFunctor >::inProgress** [protected]
- 10.86.3.2 template<typename TWorkItem, typename TFunctor> Atomic<bool> **ThreadPool< TWorkItem, TFunctor >::quit** [protected]
- 10.86.3.3 template<typename TWorkItem, typename TFunctor> std::vector<Threads::Handle*> **ThreadPool< TWorkItem, TFunctor >::threads** [protected]
- 10.86.3.4 template<typename TWorkItem, typename TFunctor> std::list<TWorkItem> **ThreadPool< TWorkItem, TFunctor >::work** [protected]
- 10.86.3.5 template<typename TWorkItem, typename TFunctor> TFunctor **ThreadPool< TWorkItem, TFunctor >::worker** [protected]
- 10.86.3.6 template<typename TWorkItem, typename TFunctor> Threads::Mutex **ThreadPool< TWorkItem, TFunctor >::workMutex** [protected]

The documentation for this class was generated from the following file:

- cputrack/threads.h

10.87 Threads Struct Reference

```
#include <threads.h>
```

Classes

- struct [Handle](#)
- struct [Mutex](#)

Public Types

- typedef void(* [ThreadEntryPoint](#)) (void *param)

Static Public Member Functions

- static DWORD WINAPI [ThreadCaller](#) (void *param)
- static [Handle](#) * [Create](#) ([ThreadEntryPoint](#) method, void *param)
- static bool [RunningVistaOrBetter](#) ()
- static void [SetBackgroundPriority](#) ([Handle](#) *thread, bool bg)
- static void [WaitAndClose](#) ([Handle](#) *h)
- static void [Sleep](#) (int ms)
- static int [GetCPUCount](#) ()

10.87.1 Member Typedef Documentation

10.87.1.1 `typedef void(* Threads::ThreadEntryPoint)(void *param)`

10.87.2 Member Function Documentation

10.87.2.1 `static Handle* Threads::Create(ThreadEntryPoint method, void * param) [inline], [static]`

```

99
100    Handle* hdl = new Handle;
101    hdl->param = param;
102    hdl->callback = method;
103    hdl->winhdl = CreateThread(0, 0, ThreadCaller, hdl, 0, &hdl->threadID);
104
105    if (!hdl->winhdl) {
106        throw std::runtime_error("Failed to create processing thread.");
107    }
108    return hdl;
109 }
```

10.87.2.2 `static int Threads::GetCPUCount() [inline], [static]`

```

138
139    // preferably
140    #ifdef WIN32
141    SYSTEM_INFO sysInfo;
142    GetSystemInfo(&sysInfo);
143    return sysInfo.dwNumberOfProcessors;
144    #else
145    return 4;
146    #endif
147 }
```

10.87.2.3 `static bool Threads::RunningVistaOrBetter() [inline], [static]`

```

112    {
113        OSVERSIONINFO v;
114        GetVersionEx(&v);
115        return v.dwMajorVersion >= 6;
116    }
```

10.87.2.4 `static void Threads::SetBackgroundPriority(Handle * thread, bool bg) [inline], [static]`

```

119    {
120        HANDLE h = (HANDLE)thread;
121        // >= Windows Vista
122        if (RunningVistaOrBetter())
123            SetThreadPriority(h, bg ? THREAD_MODE_BACKGROUND_BEGIN : THREAD_MODE_BACKGROUND_END);
124        else
125            SetThreadPriority(h, bg ? THREAD_PRIORITY_BELOW_NORMAL : THREAD_PRIORITY_NORMAL);
126    }
```

10.87.2.5 `static void Threads::Sleep(int ms) [inline], [static]`

```

134
135     ::Sleep(ms);
136 }
```

10.87.2.6 static DWORD WINAPI Threads::ThreadCaller (void * param) [inline], [static]

```

93
94     Handle* hdl = (Handle*)param;
95     hdl->callback (hdl->param);
96     return 0;
97 }
```

10.87.2.7 static void Threads::WaitAndClose (Handle * h) [inline], [static]

```

128
129     WaitForSingleObject (h->winhdl, INFINITE);
130     CloseHandle (h->winhdl);
131     delete h;
132 }
```

The documentation for this struct was generated from the following file:

- cputrack/threads.h

10.88 TImageData< T > Struct Template Reference

```
#include <QueuedTracker.h>
```

Public Member Functions

- [TImageData \(\)](#)
- [TImageData \(T *d, int w, int h\)](#)
- template<typename Ta >
[void set \(Ta *src\)](#)
- template<typename Ta >
[void set \(const TImageData< Ta > &src\)](#)
- [void copyTo \(float *dst\)](#)
- [T & at \(int x, int y\)](#)
- [T interpolate \(float x, float y, bool *outside=0\)](#)
- [T interpolate1D \(int y, float x\)](#)
- [int numPixels \(\) const](#)
- [int pitch \(\) const](#)
- [void normalize \(\)](#)
- [T mean \(\)](#)
- [T & operator\[\] \(int i\)](#)
- [void free \(\)](#)
- [void writeAsCSV \(const char *filename, const char *labels\[\]=0\)](#)

Static Public Member Functions

- [static TImageData alloc \(int w, int h\)](#)

Public Attributes

- `T * data`
- `int w`
- `int h`

10.88.1 Constructor & Destructor Documentation

10.88.1.1 template<typename T> **TImageData< T >::TImageData()** [inline]

```
82 { data=0;w=h=0; }
```

10.88.1.2 template<typename T> **TImageData< T >::TImageData(T * d, int w, int h)** [inline]

```
83 : data(d), w(w),h(h) {}
```

10.88.2 Member Function Documentation

10.88.2.1 template<typename T> static **TImageData TImageData< T >::alloc(int w, int h)** [inline], [static]

```
110 { return TImageData<T>(new T[w*h], w,h); }
```

10.88.2.2 template<typename T> **T& TImageData< T >::at(int x, int y)** [inline]

```
96 { return data[w*y+x]; }
```

10.88.2.3 template<typename T> void **TImageData< T >::copyTo(float * dst)** [inline]

```
93 {  
94     for (int i=0;i<w*h;i++) dst[i]=data[i];  
95 }
```

10.88.2.4 template<typename T> void **TImageData< T >::free()** [inline]

```
111 { if(data) delete[] data;data=0; }
```

10.88.2.5 template<typename T> **T TImageData< T >::interpolate(float x, float y, bool * outside = 0)** [inline]

```
97 { return Interpolate(data, w,h, x,y,outside); }
```

10.88.2.6 template<typename T> T **TlImageData< T >::interpolate1D(int y, float x) [inline]**

```
98 { return Interpolate1D(&data[w*y], w, x); }
```

10.88.2.7 template<typename T> T **TlImageData< T >::mean() [inline]**

```
102 {
103     T s=0.0f;
104     for(int x=0;x<w*h;x++)
105         s+=data[x];
106     return s/(w*h);
107 }
```

10.88.2.8 template<typename T> void **TlImageData< T >::normalize() [inline]**

```
101 { ::normalize(data,w,h); }
```

10.88.2.9 template<typename T> int **TlImageData< T >::numPixels() const [inline]**

```
99 { return w*h; }
```

10.88.2.10 template<typename T> T& **TlImageData< T >::operator[](int i) [inline]**

```
108 { return data[i]; }
```

10.88.2.11 template<typename T> int **TlImageData< T >::pitch() const [inline]**

```
100 { return sizeof(T)*w; } // bytes per line
```

10.88.2.12 template<typename T> template<typename Ta > void **TlImageData< T >::set(Ta * src) [inline]**

```
84 { for (int i=0;i<w*h;i++) data[i] = src[i]; }
```

10.88.2.13 template<typename T> template<typename Ta > void **TlImageData< T >::set(const **TlImageData**< Ta > & src) [inline]**

```
85 {
86     if (!data || numPixels()!=src.numPixels()) {
87         free();
88         w=src.w; h=src.h;
89         if(src.data) { data=new T[src.w*src.h]; }
90     }
91     set(src.data);
92 }
```

```
10.88.2.14 template<typename T> void TImageData< T >::writeAsCSV( const char * filename, const char * labels[] = 0
    ) [inline]

112 { WriteImageAsCSV(filename, data, w,h,labels); }
```

10.88.3 Member Data Documentation

10.88.3.1 template<typename T> T* TImageData< T >::data

10.88.3.2 template<typename T> int TImageData< T >::h

10.88.3.3 template<typename T> int TImageData< T >::w

The documentation for this struct was generated from the following files:

- cputrack/[QueuedTracker.h](#)
- cputrack/[utils.h](#)

10.89 kissfft_utils::traits< T_scalar > Struct Template Reference

```
#include <kissfft.h>
```

Public Types

- `typedef T_scalar scalar_type`
- `typedef std::complex< scalar_type > cpx_type`

Public Member Functions

- `void fill_twiddles (std::complex< T_scalar > *dst, int nfft, bool inverse) const`
- `void prepare (std::vector< std::complex< T_scalar > > &dst, int nfft, bool inverse, std::vector< int > &stageRadix, std::vector< int > &stageRemainder) const`

10.89.1 Member Typedef Documentation

10.89.1.1 template<typename T_scalar > `typedef std::complex<scalar_type> kissfft_utils::traits< T_scalar >::cpx_type`

10.89.1.2 template<typename T_scalar > `typedef T_scalar kissfft_utils::traits< T_scalar >::scalar_type`

10.89.2 Member Function Documentation

10.89.2.1 template<typename T_scalar > `void kissfft_utils::traits< T_scalar >::fill_twiddles (std::complex< T_scalar > * dst, int nfft, bool inverse) const [inline]`

```
12     {
13         T_scalar phinc = (inverse?2:-2)*acos( (T_scalar) -1) / nfft;
14         for (int i=0;i<nfft;++i)
15             dst[i] = exp( std::complex<T_scalar>(0,i*phinc) );
```

```

10.89.2.2 template<typename T_scalar> void kissfft_utils::traits< T_scalar >::prepare( std::vector< std::complex<
    T_scalar >> & dst, int nfft, bool inverse, std::vector< int > & stageRadix, std::vector< int > & stageRemainder )
const [inline]

23     {
24         std::vector<cpx_type> _twiddles;
25         _twiddles.resize(nfft);
26         fill_twiddles( &_twiddles[0],nfft,inverse );
27         dst = _twiddles;
28
29         //factorize
30         //start factoring out 4's, then 2's, then 3,5,7,9,...
31         int n= nfft;
32         int p=4;
33         do {
34             while (n % p) {
35                 switch (p) {
36                     case 4: p = 2; break;
37                     case 2: p = 3; break;
38                     default: p += 2; break;
39                 }
40                 if (p*p>n)
41                     p=n;// no more factors
42             }
43             n /= p;
44             stageRadix.push_back(p);
45             stageRemainder.push_back(n);
46         }while(n>1);
47     }

```

The documentation for this struct was generated from the following file:

- cputrack/kissfft.h

10.90 vector2< T > Struct Template Reference

```
#include <std_incl.h>
```

Public Member Functions

- [vector2 \(\)](#)
- template<typename Tx , typename Ty >
[vector2 \(Tx X, Ty Y\)](#)

Static Public Member Functions

- static [vector2 random \(vector2 center, T R\)](#)

Public Attributes

- [T x](#)
- [T y](#)

10.90.1 Constructor & Destructor Documentation

10.90.1.1 template<typename T> vector2< T >::vector2() [inline]

```
25 {x=y=0.0; }
```

10.90.1.2 template<typename T> template<typename Tx , typename Ty > **vector2< T >::vector2**(Tx X, Ty Y)
 [inline]

```
27 { x=(T)X;y=(T)Y; }
```

10.90.2 Member Function Documentation

10.90.2.1 template<typename T> static **vector2< T >::random**(**vector2< T > center**, T R) [inline],
 [static]

```
31 {
32     T ang = rand_uniform<T>() * 2 * 3.141593;
33     T r = rand_uniform<T>() * R;
34
35     return vector2(center.x + r*(T)cos(ang), center.y + r*(T)sin(ang));
36 }
```

10.90.3 Member Data Documentation

10.90.3.1 template<typename T> T **vector2< T >::x**

10.90.3.2 template<typename T> T **vector2< T >::y**

The documentation for this struct was generated from the following file:

- cputrack/std_incl.h

10.91 vector3< T > Struct Template Reference

```
#include <std_incl.h>
```

Public Member Functions

- **vector3()**
- template<typename Tx , typename Ty , typename Tz >
vector3(Tx X, Ty Y, Tz Z)
- template<typename Tc >
vector3(const **vector3**< Tc > &o)
- **vector3 operator***(const **vector3** &o) const
- **vector3 operator***(T a) const
- **vector3 operator+**(const **vector3** &o) const
- **vector3 & operator+=**(const **vector3** &o)
- **vector3 & operator-=**(const **vector3** &o)
- **vector3 operator-**(const **vector3** &o) const
- **vector3 operator-**(float a) const
- **vector3 operator+**(float a) const
- **vector3 operator-**() const
- **vector3 & operator*=**(const **vector3** &o)
- **vector3 & operator*=(**T a)
- **vector3 & operator/=(**T a)
- **vector3 operator/**(T a)
- **T length()**
- **vector2< T > xy()**

Public Attributes

- T `x`
- T `y`
- T `z`

Friends

- `vector3 operator* (T a, const vector3 &b)`
- `template<typename T>`
`vector3 operator/ (T a, vector3< T > b)`

10.91.1 Constructor & Destructor Documentation

10.91.1.1 `template<typename T> vector3< T >::vector3() [inline]`

```
44 { x=y=z=0.0f; }
```

10.91.1.2 `template<typename T> template<typename Tx , typename Ty , typename Tz > vector3< T >::vector3 (Tx X,` `Ty Y, Tz Z) [inline]`

```
46 { x=(T)X; y=(T)Y; z=(T)Z; }
```

10.91.1.3 `template<typename T> template<typename Tc > vector3< T >::vector3 (const vector3< Tc > & o)` `[inline]`

```
48 : x(o.x),y(o.y),z(o.z) {}
```

10.91.2 Member Function Documentation

10.91.2.1 `template<typename T> T vector3< T >::length () [inline]`

```
96 {
97     return sqrtf(x*x+y*y+z*z);
98 }
```

10.91.2.2 `template<typename T> vector3 vector3< T >::operator* (const vector3< T > & o) const [inline]`

```
51 {
52     return vector3(x*o.x,y*o.y,z*o.z);
53 }
```

10.91.2.3 template<typename T> vector3< T >::operator*(T a) const [inline]

```
54         {
55             return vector3(x*a,y*a,z*a);
56     }
```

10.91.2.4 template<typename T> vector3& vector3< T >::operator*=(const vector3< T > & o) [inline]

```
81         {
82             x*=o.x; y*=o.y; z*=o.z;
83             return *this;
84     }
```

10.91.2.5 template<typename T> vector3& vector3< T >::operator*=(T a) [inline]

```
85         {
86             x*=a; y*=a; z*=a;
87             return *this;
88     }
```

10.91.2.6 template<typename T> vector3< T >::operator+(const vector3< T > & o) const [inline]

```
60         {
61             return vector3(x+o.x,y+o.y,z+o.z);
62     }
```

10.91.2.7 template<typename T> vector3< T >::operator+(float a) const [inline]

```
75         {
76             return vector3(x+a,y+a,z+a);
77     }
```

10.91.2.8 template<typename T> vector3& vector3< T >::operator+=(const vector3< T > & o) [inline]

```
63         {
64             x+=o.x; y+=o.y; z+=o.z; return *this;
65     }
```

10.91.2.9 template<typename T> vector3< T >::operator-(const vector3< T > & o) const [inline]

```
69         {
70             return vector3(x-o.x,y-o.y,z-o.z);
71     }
```

10.91.2.10 template<typename T> vector3< T >::operator-(float a) const [inline]

```
72         {
73             return vector3(x-a,y-a,z-a);
74     }
```

10.91.2.11 template<typename T> vector3< T >::operator- () const [inline]

```
78     {
79         return vector3(-x,-y,-z);
80     }
```

10.91.2.12 template<typename T> vector3< T >::operator=(const vector3< T > & o) [inline]

```
66     {
67         x=o.x; y=o.y; z=o.z; return *this;
68     }
```

10.91.2.13 template<typename T> vector3< T >::operator/(T a) [inline]

```
93     {
94         return vector3(x/a,y/a,z/a);
95     }
```

10.91.2.14 template<typename T> vector3< T >::operator/=(T a) [inline]

```
89     {
90         x/=a; y/=a; z/=a;
91         return *this;
92     }
```

10.91.2.15 template<typename T> vector2<T> vector3< T >::xy() [inline]

```
105    {
106        return vector2<T>(x,y);
107    }
```

10.91.3 Friends And Related Function Documentation

10.91.3.1 template<typename T> vector3 operator*(T a, const vector3< T > & b) [friend]

```
57     {
58         return b*a;
59     }
```

10.91.3.2 template<typename T> template<typename T > vector3 operator/(T a, vector3< T > b) [friend]

```
100    {
101        return vector3<T>(a/b.x,a/b.y,a/b.z);
102    }
```

10.91.4 Member Data Documentation

10.91.4.1 template<typename T> T vector3< T >::x

10.91.4.2 template<typename T> T vector3< T >::y

10.91.4.3 template<typename T> T vector3< T >::z

The documentation for this struct was generated from the following file:

- cputrack/std_incl.h

10.92 XCor1DBuffer Class Reference

```
#include <cpu_tracker.h>
```

Public Member Functions

- [XCor1DBuffer \(int xcorw\)](#)
- void [XCorFFTHelper \(complex_t *xc, complex_t *xcr, scalar_t *result\)](#)

Public Attributes

- [kissfft< scalar_t > fft_forward](#)
- [kissfft< scalar_t > fft_backward](#)
- int [xcorw](#)

10.92.1 Constructor & Destructor Documentation

10.92.1.1 [XCor1DBuffer::XCor1DBuffer \(int xcorw \) \[inline\]](#)

```
14           : fft_forward(xcorw, false),
15           : fft_backward(xcorw, true), xcorw(xcorw)
16       {}
```

10.92.2 Member Function Documentation

10.92.2.1 void [XCor1DBuffer::XCorFFTHelper \(complex_t * xc, complex_t * xcr, scalar_t * result \)](#)

```
145 {
146     complex_t* fft_out = (complex_t*) ALLOCA(sizeof(
147     complex_t)*xcorw);
148     complex_t* fft_out_rev = (complex_t*) ALLOCA(sizeof(
149     complex_t)*xcorw);
150     fft_forward.transform(prof, fft_out);
151     fft_forward.transform(prof_rev, fft_out_rev);
152     // Multiply with conjugate of reverse
153     for (int x=0;x<xcorw;x++) {
154         fft_out[x] *= conjugate(fft_out_rev[x]);
155     }
156     fft_backward.transform(fft_out, fft_out_rev);
157     for (int x=0;x<xcorw;x++) {
158         result[x] = fft_out_rev[ (x+xcorw/2) % xcorw ].real();
159     }
160 }
```

10.92.3 Member Data Documentation

10.92.3.1 `kissfft<scalar_t> XCor1DBuffer::fft_backward`

10.92.3.2 `kissfft<scalar_t> XCor1DBuffer::fft_forward`

10.92.3.3 `int XCor1DBuffer::xcorw`

The documentation for this class was generated from the following files:

- cputrack/cpu_tracker.h
- cputrack/cpu_tracker.cpp

10.93 ZLUTParams Struct Reference

```
#include <QueuedCUDATracker.h>
```

Public Member Functions

- `CUBOTH float * GetRadialZLUT (int bead, int plane)`
- `CUBOTH int radialSteps ()`

Public Attributes

- `float minRadius`
- `float maxRadius`
- `float * zcmpwindow`
- `int angularSteps`
- `int planes`
- `cudalImageListf img`
- `float2 * trigtable`

10.93.1 Member Function Documentation

10.93.1.1 `CUBOTH float* ZLUTParams::GetRadialZLUT (int bead, int plane) [inline]`

```
41 { return img.pixelAddress(0, plane, bead); }
```

10.93.1.2 `CUBOTH int ZLUTParams::radialSteps () [inline]`

```
47 { return img.w; }
```

10.93.2 Member Data Documentation

10.93.2.1 int ZLUTParams::angularSteps

10.93.2.2 cudalimageListf ZLUTParams::img

10.93.2.3 float ZLUTParams::maxRadius

10.93.2.4 float ZLUTParams::minRadius

10.93.2.5 int ZLUTParams::planes

10.93.2.6 float2* ZLUTParams::trigtable

10.93.2.7 float* ZLUTParams::zcmpwindow

The documentation for this struct was generated from the following file:

- cudatrack/[QueuedCUDATracker.h](#)

Chapter 11

File Documentation

11.1 cputrack-test/Benchmark.cpp File Reference

```
#include "QueuedTracker.h"
#include "QueuedCPUTracker.h"
#include "SharedTests.h"
#include <functional>
#include "BenchmarkLUT.h"
#include "FisherMatrix.h"
```

Functions

- void **BenchmarkROISizes** (const char *name, int n, int MaxPixelValue, int qi_iterations, int extraFlags, float range_in_nm, float pixel_size, float lutstep, int buildLUTFlags)
- template<typename T>
void **BenchmarkConfigParamRange** (int n, T QTrkSettings::param, QTrkSettings *config, std::vector< T > param_values, const char *name, int MaxPixelValue, **vector3f** range)
- void **BenchmarkZAccuracy** (const char *name, int n, int MaxPixelValue)
- void **BenchmarkParams** ()

Variables

- const float **img_mean** = 75
- const float **img_sigma** = 1.62f
- const float **ElectronsPerBit** = **img_mean** / (**img_sigma*****img_sigma**)

11.1.1 Function Documentation

11.1.1.1 template<typename T > void BenchmarkConfigParamRange (int n, T QTrkSettings::* param, QTrkSettings * config, std::vector< T > param_values, const char * name, int MaxPixelValue, vector3f range)

```

62 {
63     std::vector<SpeedAccResult> results;
64
65     const char *lutfile = "lut000.jpg";
66     ImageData lut = ReadJPEGFile(lutfile);
67
68     for(uint i = 0; i<param_values.size();i++) {
69         QTrkSettings cfg = *config;
70         cfg.*param = param_values[i];
71
72         vector3f pos(cfg.width/2, cfg.height/2, lut.h/3);
73         std::string pfname = SPrintf("%s-%d.jpg", name, i);
74         results.push_back(SpeedAccTest (lut, &cfg, n, pos, range, pfname.c_str(), MaxPixelValue
75             ));
75     }
76     lut.free();
77
78     for (int i=0;i<results.size();i++) {
79         auto r = results[i];
80         float row[] = { param_values[i], r.acc.x, r.acc.y, r.acc.z, r.bias.x, r.bias.y, r.bias.z, r.speed,
n };
81         WriteArrayAsCSVRow(SPrintf("%s-results.txt", name).c_str(), row, sizeof(
row)/sizeof(float),i>0);
82     }
83 }
```

11.1.1.2 void BenchmarkParams ()

```

121 {
122     /*
123     - Accuracy vs ROIsize
124     - Speed vs ROIsize
125     */
126 #ifdef _DEBUG
127     int n = 50;
128 #else
129     int n = 300;
130 #endif
131
132     int mpv = 10000;
133     float pixel_size = 120, lutstep = 50;
134
135     for (int zlutbias=0;zlutbias<2;zlutbias++) {
136         float range_in_nm=0;
137         for (int bias=0;bias<2;bias++) {
138             for (int i=0;i<5;i++)
139                 BenchmarkROISizes(SPrintf("roi_qi%d_bias%d_zlutbias%d.txt",i,bias,
zlutbias).c_str(), n, mpv, i, 0, range_in_nm, pixel_size, lutstep, zlutbias ?
BUILDLUT_BIASCORRECT : 0);
140             //      for (int i=0;i<5;i++)
141             //      BenchmarkROISizes(SPrintf("roi_qi%d_bias%d_wz.txt",i,bias).c_str(), n, mpv, i,
LT_LocalizeZWeighted, range_in_nm, pixel_size, lutstep);
142             BenchmarkROISizes( SPrintf("roi_xcor_bias%d_zlutbias%d.txt", bias,
zlutbias).c_str(), n, mpv, 0, LT_XCor1D, range_in_nm, pixel_size, lutstep, zlutbias ?
BUILDLUT_BIASCORRECT : 0);
143             //      BenchmarkROISizes( SPrintf("roi_xcor_bias%d_wz.txt",bias).c_str(), n, mpv, 0, LT_XCor1D |
LT_LocalizeZWeighted, range_in_nm, pixel_size, lutstep);
144             range_in_nm = 200;
145         }
146     }
147
148     QTrkSettings basecfg;
149     basecfg.width = 80;
150     basecfg.height = 80;
151     basecfg.qi_iterations = 4;
152     basecfg.qi_roi_coverage = 1;
153     basecfg.qi_minradius=1;
154     basecfg.zlut_minradius=1;
155     basecfg.qi_radial_coverage = 2.5f;
156     basecfg.qi_angular_coverage = 0.7f;
157     basecfg.zlut_roi_coverage = 1;
158     basecfg.zlut_radial_coverage = 1.5f;
159     basecfg.com_bgcorrection = 0;
```

```

160     basecfg.qi_angstep_factor = 1.1f;
161     basecfg.zlut_angular_coverage = 0.7f;
162
163     //BenchmarkConfigParamRange (20000, &QTrkSettings::qi_iterations, &basecfg, linspace(1, 6, 6),
164     "qi_iterations_noise", mpv);
164 /*
165     BenchmarkConfigParamRange (n, &QTrkSettings::qi_radial_coverage, &basecfg, linspace(0.2f, 4.0f, 20),
166     "qi_rad_cov_noise", mpv );
167     BenchmarkConfigParamRange (n, &QTrkSettings::zlut_radial_coverage, &basecfg, linspace(0.2f, 4.0f, 20),
168     "zlut_rad_cov_noise", mpv);
169     BenchmarkConfigParamRange (n, &QTrkSettings::qi_iterations, &basecfg, linspace(1, 6, 6),
170     "qi_iterations_noise", mpv);
171     BenchmarkZAccuracy("zpos-noise.txt", n, mpv);
172
173     BenchmarkROISizes("roi-sizes.txt", n, 0);
174     BenchmarkConfigParamRange (n, &QTrkSettings::qi_radial_coverage, &basecfg, linspace(0.2f, 4.0f, 20),
175     "qi_rad_cov", 0);
176     BenchmarkConfigParamRange (n, &QTrkSettings::zlut_radial_coverage, &basecfg, linspace(0.2f, 4.0f, 20),
177     "zlut_rad_cov", 0);
178     BenchmarkConfigParamRange (n, &QTrkSettings::qi_iterations, &basecfg, linspace(1, 6, 6),
179     "qi_iterations", 0);
180     BenchmarkZAccuracy("zpos.txt", n, 0);*/
181 }
```

11.1.1.3 void BenchmarkROISizes (const char * name, int n, int MaxPixelValue, int qi_iterations, int extraFlags, float range_in_nm, float pixel_size, float lutstep, int buildLUTFlags)

```

15 {
16     std::vector<SpeedAccResult> results;
17     std::vector<int> rois;
18
19     const char *lutfile = "zrange\\exp_qi.radialzlut#169";
20     ImageData lut = ReadLUTFile(lutfile);
21
22     for (int roi=20;roi<=180;roi+=10) {
23     //for (int roi=90;roi<100;roi+=10) {
24         QTrkSettings cfg;
25         cfg.qi_angstep_factor = 1.3f;
26         cfg.qi_iterations = qi_iterations;
27         cfg.qi_angular_coverage = 0.7f;
28         cfg.qi_roi_coverage = 1;
29         cfg.qi_radial_coverage = 2.5f;
30         cfg.qi_minradius=2;
31         cfg.zlut_minradius=2;
32         cfg.zlut_angular_coverage = 0.7f;
33         cfg.zlut_roi_coverage = 1;
34         cfg.zlut_radial_coverage = 2.5f;
35         cfg.com_bgcorrection = 0;
36         cfg.xcl_profileLength = roi*0.8f;
37         cfg.xcl_profileWidth = roi*0.2f;
38         cfg.xcl_iterations = 1;
39         rois.push_back(roi);
40
41         cfg.width = roi;
42         cfg.height = roi;
43
44         vector3f pos(cfg.width/2, cfg.height/2, lut.h/2);
45         vector3f range(range_in_nm / pixel_size, range_in_nm / pixel_size, range_in_nm / lutstep);
46         results.push_back(SpeedAccTest(lut, &cfg, n, pos, range,
47             SPrintf("roi%dstesting.jpg", cfg.width).c_str(), MaxPixelValue, extraFlags, buildLUTFlags));
48         auto lr = results.back();
49         dbgprintf("ROI:%d, #QI:%d, Speed=%d img/s, Mean.X: %f. St. Dev.X: %f; Mean.Z: %f. St.
50         Dev.Z: %f\n", roi, qi_iterations, lr.speed, lr.bias.x, lr.acc.x, lr.bias.z, lr.acc.z);
51     }
52     lut.free();
53
54     for (uint i=0;i<results.size();i++) {
55         auto r = results[i];
56         float row[] = { rois[i], r.acc.x, r.acc.z, r.bias.x, r.bias.z, r.crlb.x, r.crlb.z, r.speed, n };
57         WriteArrayAsCSVRow(name, row, sizeof(row)/sizeof(float),i>0);
58     }
59 }
```

11.1.1.4 void BenchmarkZAccuracy (const char * name, int n, int MaxPixelValue)

86 {

```

87     std::vector<SpeedAccResult> results;
88     std::vector<int> zplanes;
89
90     const char *lutfile = "refbeadlut.jpg";
91     ImageData lut = ReadJPEGFile(lutfile);
92
93     for (int z=5;z<lut.h;z+=5) {
94         QTrkSettings cfg;
95         cfg.qi_angstep_factor = 2;
96         cfg.qi_iterations = 3;
97         cfg.qi_angular_coverage = 0.7f;
98         cfg.qi_roi_coverage = 1;
99         cfg.qi_radial_coverage = 1.5f;
100        cfg.zlut_angular_coverage = 0.7f;
101        cfg.zlut_roi_coverage = 1;
102        cfg.zlut_radial_coverage = 2.5f;
103
104        cfg.width = 100;
105        cfg.height = 100;
106
107        vector3f pos(cfg.width/2, cfg.height/2, z);
108        results.push_back(SpeedAccTest (lut, &cfg, n, pos, vector3f(2,2,1),
109        SPrintf("%s-zrange-%d.jpg",name,z).c_str(), MaxPixelValue));
110        zplanes.push_back(z);
111    }
112    lut.free();
113
114    for (uint i=0;i<results.size();i++) {
115        auto r = results[i];
116        float row[] = { zplanes[i], r.acc.x, r.acc.y, r.acc.z, r.bias.x, r.bias.y, r.bias.z, r.speed };
117        WriteArrayAsCSVRow(name, row, sizeof(row)/sizeof(float),i>0);
118    }

```

11.1.2 Variable Documentation

11.1.2.1 const float ElectronsPerBit = img_mean / (img_sigma*img_sigma)

11.1.2.2 const float img_mean = 75

11.1.2.3 const float img_sigma = 1.62f

11.2 cputrack-test/main.cpp File Reference

```

#include "../cputrack/std_incl.h"
#include "../cputrack/cpu_tracker.h"
#include "../cputrack/random_distr.h"
#include "../cputrack/QueuedCPUTracker.h"
#include "../cputrack/FisherMatrix.h"
#include "../cputrack/BeadFinder.h"
#include "../cputrack/LsqQuadraticFit.h"
#include "../utils/ExtractBeadImages.h"
#include "../cputrack/BenchmarkLUT.h"
#include "../cputrack/CubicBSpline.h"
#include <string>
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <sstream>
#include <time.h>
#include "testutils.h"
#include "SharedTests.h"
#include "ResultManager.h"

```

Enumerations

- enum `RWeightMode` {
 `RWNone`, `RWUniform`, `RWRadial`, `RWDerivative`,
 `RWStetson` }
- enum `Tests` { `ROIDis`, `Inter`, `Skew`, `Backg` }

Functions

- void `RescaleLUT` (`CPUTracker` *`trk`, `ImageData` *`lut`)
- void `SpeedTest` ()
- void `OnePixelTest` ()
- void `SmallImageTest` ()
- void `OutputProfileImg` ()
- void `TestBoundCheck` ()
- void `PixelationErrorTest` ()
- void `BuildConvergenceMap` (int iterations)
- void `CRP_TestGeneratedData` ()
- void `CorrectedRadialProfileTest` ()
- void `WriteRadialProf` (const char *file, `ImageData` &`d`)
- std::vector< float > `ComputeRadialWeights` (int rsteps, float minRadius, float maxRadius)
- void `TestBias` ()
- void `TestZRangeBias` (const char *name, const char *lutfile, bool normProf)
- void `TestZRange` (const char *name, const char *lutfile, int extraFlags, int clean_lut, `RWeightMode` weight←
 Mode=`RWNone`, bool biasMap=false, bool biasCorrect=false)
- void `AutoBeadFindTest` ()
- void `TestFourierLUT` ()
- void `TestFourierLUTOnDataset` ()
- void `TestZLUTAlign` ()
- void `TestQuadrantAlign` ()
- void `SimpleTest` ()
- static void `TestBSplineMax` (float maxpos)
- void `GenerateZLUTFittingCurve` (const char *lutfile)
- void `BenchmarkParams` ()
- static `SpeedAccResult` `AccBiasTest` (`ImageData` &`lut`, `QueuedTracker` *`trk`, int N, `vector3f` centerpos, `vector3f`
 range, const char *name, int MaxPixelValue, int extraFlags=0)
- void `ScatterBiasArea` (int roi, float scan_width, int steps, int samples, int qi_it, float angstep)
- void `RunCOMAndQI` (`ImageData` img, `outputter` *`output`)
- float `SkewParam` (`ImageData` img)
- void `TestROIDisplacement` (std::vector< `BeadPos` > beads, `ImageData` orilmg, `outputter` *`output`, int ROI←
 Size, int maxdisplacement=0)
- void `TestInterference` (std::vector< `BeadPos` > beads, `ImageData` orilmg, `outputter` *`output`, int ROISize,
 `vector2f` displacement=`vector2f`(60, 0))
- void `TestSkew` (std::vector< `BeadPos` > beads, `ImageData` orilmg, `outputter` *`output`, int ROISize)
- void `TestBackground` (std::vector< `BeadPos` > beads, `ImageData` orilmg, `outputter` *`output`, int ROISize)
- void `BuildZLUT` (std::string folder, `outputter` *`output`)
- void `RunZTrace` (std::string imagePath, std::string zlutPath, `outputter` *`output`)
- void `RunTest` (`Tests` test, const char *image, `outputter` *`output`, int ROISize)
- void `ManTest` ()
- void `PrintMenu` (`outputter` *`output`)
- void `SelectTests` (const char *image, int OutputMode)
- int `main` ()

Variables

- const float **ANGSTEPF** = 1.5f
- const bool **InDebugMode**

11.2.1 Enumeration Type Documentation

11.2.1.1 enum RWeightMode

Enumerator

RWNone
RWUniform
RWRadial
RWDerivative
RWStetson

```
450 { RWNone, RWUniform, RWRadial, RWDerivative,
      RWStetson };
```

11.2.1.2 enum Tests

Enumerator

ROIDis
Inter
Skew
Backg

```
1267      {
1268      ROIDis,
1269      Inter,
1270      Skew,
1271      Backg
1272 };
```

11.2.2 Function Documentation

11.2.2.1 static SpeedAccResult AccBiasTest (**ImageData & lut**, **QueuedTracker * trk**, **int N**, **vector3f centerpos**, **vector3f range**, **const char * name**, **int MaxPixelValue**, **int extraFlags = 0**) [static]

```
768 {
769     typedef QueuedTracker TrkType;
770     std::vector<vector3f> results, truepos;
771
772     int NImg=N;//std::max(1,N/20);
773     std::vector<ImageData> imgs(NImg);
774     const float R=5;
775
776     int flags= LT_LocalizeZ|LT_NormalizeProfile|extraFlags;
777     if (trk->cfg.qi_iterations>0) flags|=LT_QI;
778
779     trk->SetLocalizationMode((LocMode_t)flags);
780     Matrix3X3 fisher;
781     for (int i=0;i<NImg;i++) {
782         imgs[i]=ImageData::alloc(trk->cfg.width,trk->cfg.height);
```

```

783     vector3f pos = centerpos + range*vector3f(rand_uniform<float>()-0.5f,
784     rand_uniform<float>() -0.5f, rand_uniform<float>() -0.5f)*1;
785     GenerateImageFromLUT(&imgs[i], &lut, trk->cfg.
786     zlut_minradius, trk->cfg.zlut_maxradius, vector3f( pos.
787     x, pos.y, pos.z));
788
789     SampleFisherMatrix fm(MaxPixelValue);
790     fisher += fm.Compute(pos, vector3f(1,1,1)*0.001f, lut, trk->cfg.
791     width, trk->cfg.height, trk->cfg.zlut_minradius, trk->
792     cfg.zlut_maxradius);
793
794     LocalizationJob job(i, 0, 0, 0);
795     trk->ScheduleLocalization((uchar*)imgs[i%NImg].data, sizeof(float)*trk->
796     cfg.width, QTrkFloat, &job);
797     truepos.push_back(pos);
798 }
799 WaitForFinish(trk, N);
800
801 results.resize(trk->GetResultCount());
802 for (uint i=0;i<results.size();i++) {
803     LocalizationResult r;
804     trk->FetchResults(&r,1);
805     results[r.job.frame]=r.pos;
806 }
807
808 for (int i=0;i<NImg;i++)
809     imgs[i].free();
810
811 SpeedAccResult r;
812 r.Compute(results, [&](int index) { return truepos[index]; });
813
814 fisher *= 1.0f/NImg;
815 r.crlb = sqrt(fisher.Inverse().diag());
816 return r;
817 }
```

11.2.2.2 void AutoBeadFindTest()

```

590 {
591     auto img = ReadJPEGFile("00008153.jpg");
592     auto smp = ReadJPEGFile("00008153-s.jpg");
593     BeadFinder::Config cfg;
594     cfg.img_distance = 0.5f;
595     cfg.roi = 80;
596     cfg.similarity = 0.5;
597
598     auto results=BeadFinder::Find(&img, smp.data, &cfg);
599
600     for (uint i=0;i<results.size();i++) {
601         dbgprintf("beadpos: x=%d, y=%d\n", results[i].x, results[i].y);
602         img.at(results[i].x+cfg.roi/2, results[i].y+cfg.roi/2) = 1.0f;
603     }
604     dbgprintf("%d beads total\n", results.size());
605
606     FloatToJPEGFile("autobeadfind.jpg", img.data, img.w, img.h);
607
608     img.free();
609     smp.free();
610 }
```

11.2.2.3 void BenchmarkParams()

```

121 {
122     /*
123     - Accuracy vs ROIsize
124     - Speed vs ROIsize
125     */
126 #ifdef _DEBUG
127     int n = 50;
128 #else
129     int n = 300;
130 #endif
131 }
```

```

132     int mpv = 10000;
133     float pixel_size = 120, lutstep = 50;
134
135     for (int zlutbias=0;zlutbias<2;zlutbias++) {
136         float range_in_nm=0;
137         for (int bias=0;bias<2;bias++) {
138             for (int i=0;i<5;i++)
139                 BenchmarkROISizes(SPrintf("roi_qi%d_bias%d_zlutbias%d.txt",i,bias,
zlutbias).c_str(), n, mpv, i, 0, range_in_nm, pixel_size, lutstep, zlutbias ?
BUILDLUT_BIASCORRECT : 0);
140             // for (int i=0;i<5;i++)
141             // BenchmarkROISizes(SPrintf("roi_qi%d_bias%d_wz.txt",i,bias).c_str(), n, mpv, i,
LT_LocalizeZWeighted, range_in_nm, pixel_size, lutstep);
142             BenchmarkROISizes( SPrintf("roi_xcor_bias%d_zlutbias%d.txt", bias,
zlutbias).c_str(), n, mpv, 0, LT_XCor1D, range_in_nm, pixel_size, lutstep, zlutbias ?
BUILDLUT_BIASCORRECT : 0);
143             // BenchmarkROISizes( SPrintf("roi_xcor_bias%d_wz.txt",bias).c_str(), n, mpv, 0, LT_XCor1D |
LT_LocalizeZWeighted, range_in_nm, pixel_size, lutstep);
144             range_in_nm = 200;
145         }
146     }
147
148     QTrkSettings basecfg;
149     basecfg.width = 80;
150     basecfg.height = 80;
151     basecfg.qi_iterations = 4;
152     basecfg.qi_roi_coverage = 1;
153     basecfg.qi_minradius=1;
154     basecfg.zlut_minradius=1;
155     basecfg.qi_radial_coverage = 2.5f;
156     basecfg.qi_angular_coverage = 0.7f;
157     basecfg.zlut_roi_coverage = 1;
158     basecfg.zlut_radial_coverage = 1.5f;
159     basecfg.com_bgcorrection = 0;
160     basecfg.qi_angstep_factor = 1.1f;
161     basecfg.zlut_angular_coverage = 0.7f;
162
163     //BenchmarkConfigParamRange (20000, &QTrkSettings::qi_iterations, &basecfg, linspace(1, 6, 6),
"qi_iterations_noise", mpv);
164 /*
165     BenchmarkConfigParamRange (n, &QTrkSettings::qi_radial_coverage, &basecfg, linspace(0.2f, 4.0f, 20),
"qi_rad_cov_noise", mpv );
166     BenchmarkConfigParamRange (n, &QTrkSettings::zlut_radial_coverage, &basecfg, linspace(0.2f, 4.0f, 20),
"zlut_rad_cov_noise", mpv );
167     BenchmarkConfigParamRange (n, &QTrkSettings::qi_iterations, &basecfg, linspace(1, 6, 6),
"qi_iterations_noise", mpv );
168     BenchmarkZAccuracy("zpos-noise.txt", n, mpv);
169
170     BenchmarkROISizes("roi-sizes.txt", n, 0);
171     BenchmarkConfigParamRange (n, &QTrkSettings::qi_radial_coverage, &basecfg, linspace(0.2f, 4.0f, 20),
"qi_rad_cov", 0);
172     BenchmarkConfigParamRange (n, &QTrkSettings::zlut_radial_coverage, &basecfg, linspace(0.2f, 4.0f, 20),
"zlut_rad_cov", 0);
173     BenchmarkConfigParamRange (n, &QTrkSettings::qi_iterations, &basecfg, linspace(1, 6, 6),
"qi_iterations", 0);
174     BenchmarkZAccuracy("zpos.txt", n, 0);*/
175 }
```

11.2.2.4 void BuildConvergenceMap (int *iterations*)

```

257 {
258     int W=80, H=80;
259     char* data=new char[W*H];
260     FILE* f=fopen("singlebead.bin", "rb");
261     fread(data,1,80*80,f);
262     fclose(f);
263
264     float testrange=20;
265     int steps=100;
266     float step=testrange/steps;
267     vector2f errXCor, errQI;
268     CPUTracker trk(W,H,40);
269
270     // Get a good starting estimate
271     trk.SetImage8Bit((uchar*)data,W);
272     vector2f com = trk.ComputeMeanAndCOM();
273     bool boundaryHit;
274     vector2f cmp = trk.ComputeQI(com,8,80,64,ANGSTEPF,2,25,boundaryHit);
275
276     float *xcorErrMap = new float[steps*steps];
277     float *qiErrMap = new float[steps*steps];
278 }
```

```

279     for (int y=0;y<steps;y++) {
280         for (int x=0;x<steps;x++) {
281             {
282                 vector2f initial (cmp.x+step*(x-steps/2), cmp.y+step*(y-steps/2) );
283                 vector2f xcor = trk.ComputeXCorInterpolated(initial, iterations, 64, boundaryHit);
284                 vector2f qi = trk.ComputeQI(initial, iterations, 80, 64,
285 ANGSTEPF,2,30,boundaryHit);
286 
287                 errXCor.x += fabs(xcor.x-cmp.x);
288                 errXCor.y += fabs(xcor.y-cmp.y);
289                 xcorErrMap[y*steps+x] = distance(xcor,cmp);
290 
291                 errQI.x += fabs(qi.x-cmp.x);
292                 errQI.y += fabs(qi.y-cmp.y);
293                 qiErrMap[y*steps+x] = distance(qi,cmp);
294             }
295             dbgprintf("y=%d\n", y);
296         }
297 
298         WriteImageAsCSV(SPrintf("xcor-err-i%d.csv", iterations).c_str(), xcorErrMap,
299 steps,steps);
300         WriteImageAsCSV(SPrintf("qi-err-i%d.csv", iterations).c_str(), qiErrMap, steps,
301 steps);
302         delete[] qiErrMap;
303         delete[] xcorErrMap;
304         delete[] data;
305     }

```

11.2.2.5 void BuildZLUT (std::string folder, outputter * output)

```

998 {
999     int ROISize = 100;
1000    std::vector<BeadPos> beads = read_beadlist(SPrintf("%sbeadlist.txt",folder.c_str()));
1001
1002
1003    int numImgInStack = 1218;
1004    int numPositions = 1001; // 10nm/frame
1005    float range = 10.0f; // total range 25.0 um -> 35.0 um
1006    float umPerImg = range/numImgInStack;
1007
1008    QTrkComputedConfig cfg;
1009    cfg.width=cfg.height = ROISize;
1010    cfg.qi_angstep_factor = 1;
1011    cfg.qi_iterations = 6;
1012    cfg.qi_angular_coverage = 0.7f;
1013    cfg.qi_roi_coverage = 1;
1014    cfg.qi_radial_coverage = 1.5f;
1015    cfg.qi_minradius=0;
1016    cfg.zlut_minradius=0;
1017    cfg.zlut_angular_coverage = 0.7f;
1018    cfg.zlut_roi_coverage = 1;
1019    cfg.zlut_radial_coverage = 1.5f;
1020    cfg.zlut_minradius = 0;
1021    cfg.qi_minradius = 0;
1022    cfg.com_bgcorrection = 0;
1023    cfg.xcl_profileLength = ROISize*0.8f;
1024    cfg.xcl_profileWidth = ROISize*0.2f;
1025    cfg.xcl_iterations = 1;
1026    cfg.Update();
1027    cfg.WriteToFile();
1028
1029    int zplanes = 50;
1030
1031    QueuedCPUTracker* qtrk = new QueuedCPUTracker(cfg);
1032    qtrk->SetLocalizationMode(LT_NormalizeProfile |
LT_QI);
1033    qtrk->SetRadialZLUT(0, beads.size(), zplanes);
1034    qtrk->BeginLUT(0);
1035
1036    int pxPerBead = ROISize*ROISize;
1037    int memSizePerBead = pxPerBead*sizeof(float);
1038    int startFrame = 400;
1039    for(int plane = 0; plane < zplanes; plane++){
1040        output->outputString(SPrintf("Frame %d/%d",plane+1,zplanes),true);
1041        int frameNum = startFrame+(int)(numImgInStack-startFrame)*((float)plane/zplanes);
1042        std::string file = SPrintf("%s\\img%05d.jpg",folder.c_str(),frameNum);
1043
1044        ImageData frame = ReadJPEGFile(file.c_str());
1045        float* data = new float[beads.size()*pxPerBead];

```

```

1047
1048     for(uint ii = 0; ii < beads.size(); ii++){
1049         vector2f pos;
1050         pos.x = beads.at(ii).x - ROISize/2;
1051         pos.y = beads.at(ii).y - ROISize/2;
1052         ImageData crop = CropImage(frame, pos.x, pos.y, ROISize, ROISize);
1053         //output->outputImage(crop, SPrintf("%d-%05d", ii, plane));
1054         memcpy(data+ii*pxPerBead, crop.data, memSizePerBead);
1055         crop.free();
1056     }
1057
1058     /*
1059     // To verify seperate frame bead stack generation
1060     output->newFile(SPrintf("data-plane-%d", plane));
1061     output->outputArray(data, beads.size()*pxPerBead);
1062
1063     ImageData allBeads = ImageData(data, ROISize, ROISize*beads.size());
1064     output->outputImage(allBeads, SPrintf("allBeads-%05d", frameNum)); //*/
1065
1066     qtrk->BuildLUT(data, sizeof(float)*ROISize, QTrkFloat, plane);
1067
1068     frame.free();
1069     delete[] data;
1070 }
1071
1072 qtrk->FinalizeLUT();
1073
1074 for(int ii = 0; ii < beads.size(); ii++){
1075     ImageData lut = ImageData::alloc(cfg.
1076     zlut_radialsteps, zplanes);
1077     memcpy(lut.data, qtrk->GetZLUTByIndex(ii), cfg.
1078     zlut_radialsteps*zplanes*sizeof(float));
1079     //output->outputImage(lut, SPrintf("lut%03d,%d", beads.at(ii).x, beads.at(ii).y));
1080     output->outputImage(lut, SPrintf("lut%03d", ii));
1081     lut.free();
1082 }
1083
1084 qtrk->Flush();
1085 delete qtrk;
1086 }
```

11.2.2.6 std::vector<float> ComputeRadialWeights (int rsteps, float minRadius, float maxRadius)

```

366 {
367     std::vector<float> wnd(rsteps);
368     for(int x=0;x<rsteps;x++)
369         wnd[x]=Lerp(minRadius, maxRadius, x/(float)rsteps) / (0.5f * (minRadius+maxRadius));
370     return wnd;
371 }
```

11.2.2.7 void CorrectedRadialProfileTest()

```

320 {
321     // read image
322     const char* imgname = "SingleBead.jpg";
323     ImageData img = ReadJPEGFile(imgname);
324
325     // localize
326     CPUTracker trk(img.w, img.h);
327     trk.SetImageFloat(img.data);
328     vector2f com = trk.ComputeMeanAndCOM();
329     bool boundaryHit;
330     vector2f qi = trk.ComputeQI(com, 4, 64, 64, ANGSTEPF, 1, 30, boundaryHit);
331     dbgprintf("%s: COM: %f, %f. QI: %f, %f\n", imgname, com.x, com.y, qi.
332     x, qi.y);
333     std::vector<float> angularProfile(128);
334     float asym = trk.ComputeAsymmetry(qi, 64, angularProfile.size(), 1, 30, &angularProfile[0]);
335 // ComputeAngularProfile(&angularProfile[0], 64, angularProfile.size(), 1, 30, qi, &img, trk.mean);
336     WriteImageAsCSV("angprof.csv", &angularProfile[0], angularProfile.size(), 1);
337     dbgprintf("Asymmetry value: %f\n", asym);
338     std::vector<float> crp(128);
339     float* crpmap = new float[angularProfile.size()*crp.size()];
340     ComputeCRP(&crp[0], crp.size(), angularProfile.size(), 1, 30, qi, &img, 0.0f);
341     WriteImageAsCSV("crpmap.csv", crpmap, crp.size(), angularProfile.size());
342     delete[] crpmap;
343     delete[] img.data;
```

```

344     //CRP_TestGeneratedData();
345
346
347 //  GenerateImageFromLUT(ImageData(img,w,h), ImageData
348 }
```

11.2.2.8 void CRP_TestGeneratedData()

```

307 {
308     int w=64,h=64;
309     float* img = new float[w*h];
310     const char* lutname = "LUTexample25X.jpg";
311     std::vector<uchar> lutdata = ReadToByteBuffer(lutname);
312
313     int lutw,luth;
314     uchar* lut;
315     ReadJPEGfile(&lutdata[0], lutdata.size(), &lut, &lutw, &luth);
316     delete[] img;
317 }
```

11.2.2.9 void GenerateZLUTFittingCurve(const char * lutfile)

```

729 {
730     QTrkSettings settings;
731     settings.width = settings.height = 80;
732
733     QueuedCPUTracker qt(settings);
734     ImageData lut = ReadJPEGfile(lutfile);
735     ImageData nlut;
736     ResampleLUT(&qt, &lut, lut.h, &nlut);
737
738     CPUTracker trk(settings.width,settings.height);
739
740     ImageData smp = ImageData::alloc(settings.width,settings.
height);
741
742     trk.SetRadialZLUT(nlut.data, nlut.h, nlut.w, 1, qt.cfg.zlut_minradius, qt.cfg.zlut_maxradius,
false, false);
743
744     int N=8;
745     for (int z=0;z<6;z++) {
746         vector3f pos(settings.width/2,settings.height/2, nlut.
h * (1+z) / (float)N + 0.123f);
747         GenerateImageFromLUT(&smp, &nlut, qt.cfg.zlut_minradius, qt.cfg.zlut_maxradius,
pos);
748         ApplyPoissonNoise(smp, 10000);
749         WriteJPEGfile( SPrintf("zlutfitcurve-smpimg-z%d.jpg", z).c_str(), smp);
750         trk.SetImageFloat(smp.data);
751         std::vector<float> profile(qt.cfg.zlut_radialsteps), cmpProf(nlut.h), fitted(nlut.
h);
752         trk.ComputeRadialProfile(&profile[0], qt.cfg.zlut_radialsteps, qt.cfg.zlut_angularsteps, qt.cfg.
zlut_minradius, qt.cfg.zlut_maxradius, pos.xy(), false);
753         trk.LUTProfileCompare(&profile[0], 0, &cmpProf[0],
CPUTracker::LUTProfMaxQuadraticFit, &fitted[0]);
754
755         WriteArrayAsCSVRow("zlutfitcurve-profile.txt", &profile[0], profile.size(), z>0);
756         WriteArrayAsCSVRow("zlutfitcurve-cmpprof.txt", &cmpProf[0], cmpProf.size(), z>0);
757         WriteArrayAsCSVRow("zlutfitcurve-fitted.txt", &fitted[0], fitted.size(), z>0);
758     }
759
760     smp.free();
761     nlut.free();
762     lut.free();
763 }
```

11.2.2.10 int main()

```

1510 {
1511 #ifdef _DEBUG
1512 //  Matrix3X3::test();
1513 #endif
```

```

1514     SelectTests("D:\\TestImages\\img00095.jpg", Files+Images);
1515 // ManTest();
1516
1517 // SimpleTest();
1518
1519 // GenerateZLUTFittingCurve("lut000.jpg");
1520
1521 /*TestBias();
1522 TestZRangeBias("ref169-norm", "zrange\\exp_qi.radialzlut#169", true);
1523 TestZRangeBias("ref169-raw", "zrange\\exp_qi.radialzlut#169", false);
1524
1525 // SmallROITest("lut000.jpg");
1526
1527 //TestZRange("lut227-ref","lut227.jpg", 0, 0, RWStetson);
1528 TestZRange("zrange\\lut169ref-biasmap-c","zrange\\exp_qi.radialzlut#169", 0, 0, RWStetson, true, true);
1529 TestZRange("zrange\\lut169ref-biasmap","zrange\\exp_qi.radialzlut#169", 0, 0, RWStetson, true, false);
1530 TestZRange("zrange\\lut169ref","zrange\\exp_qi.radialzlut#169", 0, 0, RWStetson, false, false);
1531 TestZRange("zrange\\lut169ref-c","zrange\\exp_qi.radialzlut#169", 0, 0, RWStetson, false, true);
1532 TestZRange("zrange\\lut013tether","zrange\\exp_qi.radialzlut#13", 0, 0, RWStetson, false, false);
1533 TestZRange("zrange\\lut013tether-c","zrange\\exp_qi.radialzlut#13", 0, 0, RWStetson, false, true);*/
1534 /*
1535 TestZRange("zrange\\longlут1-c","zrange\\long.radialzlut#1", 0,0, RWStetson, false, true);
1536 TestZRange("zrange\\longlут1","zrange\\long.radialzlut#1", 0,0, RWStetson, false, false);
1537 TestZRange("zrange\\longlут3-c","zrange\\long.radialzlut#3", 0,0, RWStetson, false, true);
1538 TestZRange("zrange\\longlут3","zrange\\long.radialzlut#3", 0,0, RWStetson, false, false);
1539 */
1540 //TestZRange("cleanlut1", "lut000.jpg", LT_LocalizeZWeighted, 0);
1541 //TestZRange("cleanlut1", "lut000.jpg", LT_LocalizeZWeighted, 1);
1542 //TestZRange("cleanlut10", "lut10.jpg", LT_LocalizeZWeighted, 1);
1543 //TestZRange("cleanlut10", "lut10.jpg", LT_LocalizeZWeighted, 1);
1544
1545 // BenchmarkParams();
1546 // int N=50;
1547 /*ScatterBiasArea(80, 4, 100, N, 3, 1);
1548 ScatterBiasArea(80, 4, 100, N, 4, 1);
1549 ScatterBiasArea(80, 4, 100, N, 1, 1);
1550 ScatterBiasArea(80, 4, 100, N, 2, 1);
1551 ScatterBiasArea(80, 4, 100, N, 0, 1);
1552 ScatterBiasArea(80, 4, 100, N, -1, 1);
1553 */
1554 /*
1555 ImageData img=ReadLUTFile("lut000.jpg");
1556 img.mean();
1557
1558 TestZRange("rbin1x", "1x.radialzlut#4", 0, 0, RWUniform);
1559 TestZRange("rbin1x", "1x.radialzlut#4", 0, 0, RWRadial);
1560 TestZRange("rbin1x", "1x.radialzlut#4", 0, 0, RDDerivative);
1561
1562 TestZRange("rbin10x", "10x.radialzlut#4", 0, 0, RWUniform);
1563 TestZRange("rbin10x", "10x.radialzlut#4", 0, 0, RWRadial);
1564 TestZRange("rbin10x", "10x.radialzlut#4", 0, 0, RDDerivative);
1565 */
1566 // QTrkTest();
1567 // TestCMOSNoiseInfluence<QueuedCPUTracker>("lut000.jpg");
1568
1569 //AutoBeadFindTest();
1570 // Gauss2DTest<QueuedCPUTracker>();
1571
1572 //SpeedTest();
1573 //PixelationErrorTest();
1574 //ZTrackingTest();
1575 //Test2DTracking();
1576 //TestBoundCheck();
1577 //QTrkTest();
1578 //for (int i=1;i<8;i++)
1579 // BuildConvergenceMap(i);
1580
1581 //TestFourierLUT();
1582 // TestQuadrantAlign();
1583 // TestZLUTAlign();
1584 // TestImageLUT();
1585 // TestBuildRadialZLUT<QueuedCPUTracker>( "lut000.jpg" );
1586 // TestImageLUT();
1587
1588 //CorrectedRadialProfileTest();
1589
1590 //system("pause");
1591 return 0;
1592 }

```

11.2.2.11 void ManTest()

1314 {

```

1315     int ROISize = 101;
1316     float displacement = 0;
1317     float skewFact = 0.0f;
1318     float bgCorr = 2;
1319     int imgSel = 0;
1320
1321     char inChar;
1322     do
1323     {
1324         char selChar;
1325         do{
1326             printf_s("Current settings: image %s\nROI %d, skewFact %f, displacement %f, bgCorr %f\n\n", (
1327             imgSel == 0)? "generated":"CroppedBead.jpg", ROISize, skewFact, displacement, bgCorr);
1328             std::cout << "What setting to change? (? for list, 0 to run)\n";
1329             std::cin >> selChar;
1330             switch(selChar){
1331                 case '?':
1332                     printf("i: imgSel (0: Generated, 1: CroppedBead.jpg)\nr: ROISize\nb: COM Background
correction\n");
1333                     printf("\nOnly for generated image:\n\tts: skewFact\n\ttd: displacement\n");
1334                     break;
1335                 case 'i':
1336                     std::cin >> imgSel;
1337                     break;
1338                 case 'r':
1339                     std::cin >> ROISize;
1340                     break;
1341                 case 's':
1342                     std::cin >> skewFact;
1343                     break;
1344                 case 'd':
1345                     std::cin >> displacement;
1346                     break;
1347                 case 'b':
1348                     std::cin >> bgCorr;
1349                     break;
1350                 default:
1351                     break;
1352             }
1353             std::cout << std::endl;
1354         } while(selChar != '0');
1355         ImageData img;
1356         if(imgSel == 1){
1357             ImageData imgRaw = ReadJPEGFile("D:\\TestImages\\CroppedBead.jpg");
1358             ROISize = imgRaw.w;
1359             img = SkewImage(imgRaw,skewFact);
1360             img.normalize();
1361             imgRaw.free();
1362         } else {
1363             ImageData imgRaw = ImageData::alloc(ROISize,ROISize);
1364             GenerateTestImage(imgRaw,(float)ROISize/2+displacement,(float)ROISize/2+
displacement,5,0);
1365             img = SkewImage(imgRaw,skewFact);
1366             img.normalize();
1367             imgRaw.free();
1368         }
1369         FloatToJPEGFile("D:\\TestImages\\test.jpg",img.data,ROISize,ROISize);
1370
1371         QTrkComputedConfig cfg;
1372         cfg.width = cfg.height = ROISize;
1373         cfg.qi_angstep_factor = 1;
1374         cfg.qi_iterations = 10;
1375         cfg.qi_angular_coverage = 0.7f;
1376         cfg.qi_roi_coverage = 1.0f;
1377         cfg.qi_radial_coverage = 1.5f;
1378         cfg.qi_minradius = 0;
1379         cfg.zlut_minradius = 2;
1380         cfg.zlut_radial_coverage = 2;
1381         cfg.zlut_angular_coverage = 0.7f;
1382         cfg.zlut_roi_coverage = 1;
1383         cfg.com_bgcorrection = bgCorr;
1384         cfg.xcl_profileLength = ROISize*0.8f;
1385         cfg.xcl_profileWidth = ROISize*0.2f;
1386         cfg.xcl_iterations = 4;
1387         cfg.testRun = true;
1388
1389         cfg.Update();
1390
1391         QueuedCPUTracker* qtrk = new QueuedCPUTracker(cfg);
1392         qtrk->SetLocalizationMode(LT_QI | LT_LocalizeZ |
LT_NormalizeProfile);
1393         LocalizationJob job(0, 0, 0, 0, 0);
1394         ROIPosition pos;
1395         pos.x = 0;
1396         pos.y = 0;
1397         int queued = qtrk->ScheduleFrame(img.data,sizeof(float)*img.
w,img.w,img.h,&pos,1,QTrkFloat,&job);

```

```

1397     printf("q: %d\n", queued);
1398     while(qtrk->GetQueueLength() != 0);
1399     while(qtrk->GetResultCount() != 0) {
1400         LocalizationResult lr;
1401         qtrk->FetchResults(&lr,1);
1402
1403         printf("%f %f %f %f\n", lr.firstGuess.x, lr.firstGuess.
1404               y, lr.pos.x, lr.pos.y);
1405
1406         float* prof = ALLOCA_ARRAY(float, cfg.zlut_radialsteps);
1407         bool boundaryHit;
1408         ImageData imgData (img.data, ROISize, ROISize);
1409         ComputeRadialProfile(prof, cfg.zlut_radialsteps, cfg.
1410             zlut_angularsteps, cfg.zlut_minradius, cfg.
1411             zlut_maxradius, lr.pos.xy(), &imgData, lr.imageMean, false);
1412         WriteArrayAsCSVRow("D:\\TestImages\\RadialProfile.txt", prof, cfg.
1413             zlut_radialsteps, false);
1414         ComputeRadialProfile(prof, cfg.zlut_radialsteps, cfg.
1415             zlut_angularsteps, cfg.zlut_minradius, cfg.
1416             zlut_maxradius, lr.pos.xy(), &imgData, lr.imageMean, true);
1417         WriteArrayAsCSVRow("D:\\TestImages\\RadialProfile.txt", prof, cfg.
1418             zlut_radialsteps, true);
1419     }
1420 }
```

11.2.2.12 void OnePixelTest()

```

132 {
133     CPUTracker* tracker = new CPUTracker(32,32, 16);
134
135     tracker->GetPixel(15,15) = 1;
136     dbgout(SPrintf("Pixel at 15,15\n"));
137     vector2f com = tracker->ComputeMeanAndCOM();
138     dbgout(SPrintf("COM: %f,%f\n", com.x, com.y));
139
140     vector2f initial(15,15);
141     bool boundaryHit = false;
142     vector2f xcor = tracker->ComputeXCorInterpolated(initial,2, 16,
143             boundaryHit);
144     dbgout(SPrintf("XCor: %f,%f\n", xcor.x, xcor.y));
145
146     assert(xcor.x == 15.0f && xcor.y == 15.0f);
147     delete tracker;
```

11.2.2.13 void OutputProfileImg()

```

179 {
180     CPUTracker *tracker = new CPUTracker(128,128, 16);
181     bool boundaryHit;
182
183     for (int i=0;i<10;i++) {
184         float xp = tracker->GetWidth()/2+(rand_uniform<float>() - 0.5) * 20;
185         float yp = tracker->GetHeight()/2+(rand_uniform<float>() - 0.5) * 20;
186
187         GenerateTestImage(ImageData(tracker->srcImage, tracker->
188             GetWidth(), tracker->GetHeight()), xp, yp, 1, 0.0f);
189
190         vector2f com = tracker->ComputeMeanAndCOM();
191         dbgout(SPrintf("COM: %f,%f\n", com.x-xp, com.y-yp));
192
193         vector2f initial = com;
194         boundaryHit=false;
195         vector2f xcor = tracker->ComputeXCorInterpolated(initial, 3, 16,
196             boundaryHit);
197         dbgprintf("XCor: %f,%f. Err: %d\n", xcor.x-xp, xcor.y-yp, boundaryHit);
198
199         boundaryHit=false;
200         vector2f qi = tracker->ComputeQI(initial, 3, 64, 32,
ANGSTEPF, 1, 10, boundaryHit);
```

```

199     dbgprintf("QI: %f,%f. Err: %d\n", qi.x-xp, qi.y-yp, boundaryHit);
200 }
201
202 delete tracker;
203 }
```

11.2.2.14 void PixelationErrorTest()

```

233 {
234     CPUTracker *tracker = new CPUTracker(128,128, 64);
235
236     float X = tracker->GetWidth()/2;
237     float Y = tracker->GetHeight()/2;
238     int N = 20;
239     for (int x=0;x<N;x++) {
240         float xpos = X + 2.0f * x / (float)N;
241         GenerateTestImage(ImageData(tracker->srcImage, tracker->
242             GetWidth(), tracker->GetHeight()), xpos, X, 1, 0.0f);
243
244         vector2f com = tracker->ComputeMeanAndCOM();
245         //dbgout(SPrintf("COM: %f,%f\n", com.x, com.y));
246
247         vector2f initial(X,Y);
248         bool boundaryHit = false;
249         vector2f xcorInterp = tracker->ComputeXCorInterpolated(initial, 3, 3
2, boundaryHit);
250         vector2f qipos = tracker->ComputeQI(initial, 3, tracker->
251             GetWidth(), 128, 1, 2.0f, tracker->GetWidth()/2-10, boundaryHit);
252         dbgprintf("xpos:%f, COM err: %f, XCorInterp err: %f. QI err: %f\n", xpos, com.
253         x-xpos, xcorInterp.x-xpos, qipos.x-xpos);
254 }
```

11.2.2.15 void PrintMenu(outputter * output)

```

1423 {
1424     output->outputString("0. Quit",true);
1425     output->outputString("1. ROI Displacement",true);
1426     output->outputString("2. Interference",true);
1427     output->outputString("3. Skew",true);
1428     output->outputString("4. Background",true);
1429     output->outputString("5. Build ZLUTs",true);
1430     output->outputString("6. Run Trace",true);
1431     output->outputString("m. Manual",true);
1432     output->outputString("r. Change ROI size",true);
1433     output->outputString("?. Menu",true);
1434 }
```

11.2.2.16 void RescaleLUT(CPUTracker * trk, ImageData * lut)

```

32 {
33
34 }
```

11.2.2.17 void RunCOMAndQI (*ImageData img, outputter * output*)

```

876
877     char buf[256];
878     CPUTracker trk(img.w, img.h);
879     trk.SetImageFloat(img.data);
880     double t = GetPreciseTime();
881     vector2f com = trk.ComputeMeanAndCOM();
882     t = GetPreciseTime() - t;
883     //float asym = trk.ComputeAsymmetry(com, 64, 64, 5, 50, dstAngProf);
884     sprintf(buf,"%f %f %f",com.x,com.y,t);
885     output->outputString(buf);
886
887     vector2f initial(com.x, com.y);
888
889     bool boundaryHit = false;
890     for(int qi_iterations = 1; qi_iterations < 10; qi_iterations++){
891         t = GetPreciseTime();
892         vector2f qi = trk.ComputeQI(initial, qi_iterations, 64, 16,
893             ANGSTEPF, 5, 50, boundaryHit);
894         t = GetPreciseTime() - t;
895         //float asym = trk.ComputeAsymmetry(qi, 64, 64, 5, 50, dstAngProf);
896         sprintf(buf,"%f %f %f",qi.x,qi.y,t);
897         output->outputString(buf);
898         boundaryHit = false;
899     }

```

11.2.2.18 void RunTest (*Tests test, const char * image, outputter * output, int ROISize*)

```

1275 {
1276     ImageData source = ReadJPEGFile(image);
1277     std::vector<BeadPos> beads = read_beadlist("D:\\TestImages\\beadlist.txt");
1278     output->newFile("TestInfo","a");
1279
1280     if(test == ROIDis)
1281         output->outputString("ROI Displacement test");
1282     else if(test == Inter)
1283         output->outputString("Interference test");
1284     else if(test == Skew)
1285         output->outputString("Skew test");
1286     else if(test == Backg)
1287         output->outputString("Background test");
1288     output->outputString(Sprintf("Image %s\\nBeadlist D:\\TestImages\\beadlist.txt\\n
1289     NumBeads %d\\nROISize %d",image,beads.size(),ROISize));
1290
1291     double t0 = GetPreciseTime();
1292
1293     switch(test) {
1294     case ROIDis:
1295         TestROIDisplacement(beads,source,output,ROISize);
1296         break;
1297     case Inter:
1298         TestInterference(beads,source,output,ROISize);
1299         break;
1300     case Skew:
1301         TestSkew(beads,source,output,ROISize);
1302         break;
1303     case Backg:
1304         TestBackground(beads,source,output,ROISize);
1305         break;
1306     }
1307
1308     double t1 = GetPreciseTime();
1309     output->newFile("TestInfo","a");
1310     output->outputString(Sprintf("Duration %f\\n",t1-t0));
1311     source.free();

```

11.2.2.19 void RunZTrace (*std::string imagePath, std::string zlutPath, outputter * output*)

```

1087 {
1088     int ROISize = 100;
1089     std::vector<BeadPos> beads = read_beadlist(SPrintf("%sbeadlist.txt",imagePath.c_str()));
1090     //std::vector<BeadPos> beads = read_labview_beadlist(SPrintf("%sbeadlist.txt",imagePath.c_str()));

```

```

1091     if(beads.size() == 0){
1092         output->outputString("Empty beadlist!",true);
1093         return;
1094     }
1095
1096     QTrkComputedConfig cfg;
1097     /*
1098     output->outputString("Enter used ZLUT settings.",true);
1099     output->outputString(SPrintf("ROI size (currently %d)",ROISize),true);
1100     std::cin >> ROISize;
1101     output->outputString("ZLUT radial sample density (default 1)",true);
1102     std::cin >> cfg.zlut_radial_coverage;
1103     output->outputString("ZLUT minradius (default 2)",true);
1104     std::cin >> cfg.zlut_minradius;
1105     output->outputString("ZLUT ROI coverage (default 1)",true);
1106     std::cin >> cfg.zlut_roi_coverage;
1107     output->outputString("ZLUT angular coverage (default 0.7)",true);
1108     std::cin >> cfg.zlut_angular_coverage;
1109     */
1110     /*
1111     cfg.width = cfg.height = ROISize;
1112     cfg.qi_angstep_factor = 1;
1113     cfg.qi_iterations = 6;
1114     cfg.qi_angular_coverage = 0.7f;
1115     cfg.qi_roi_coverage = 1;
1116     cfg.qi_radial_coverage = 1.5f;
1117     cfg.qi_minradius = 0;
1118     cfg.zlut_minradius = 2;
1119     cfg.zlut_radial_coverage = 2;
1120     cfg.zlut_angular_coverage = 0.7f;
1121     cfg.zlut_roi_coverage = 1;
1122     cfg.com_bgcorrection = 0;
1123     cfg.xcl_profileLength = ROISize*0.8f;
1124     cfg.xcl_profileWidth = ROISize*0.2f;
1125     cfg.xcl_iterations = 1;
1126     cfg.testRun = true;
1127     */
1128     cfg.width=cfg.height = ROISize;
1129     cfg.qi_angstep_factor = 1;
1130     cfg.qi_iterations = 6;
1131     cfg.qi_angular_coverage = 0.7f;
1132     cfg.qi_roi_coverage = 1;
1133     cfg.qi_radial_coverage = 1.5f;
1134     cfg.qi_minradius=0;
1135     cfg.zlut_minradius=0;
1136     cfg.zlut_angular_coverage = 0.7f;
1137     cfg.zlut_roi_coverage = 1;
1138     cfg.zlut_radial_coverage = 1.5f;
1139     cfg.zlut_minradius = 0;
1140     cfg.qi_minradius = 0;
1141     cfg.com_bgcorrection = 0;
1142     cfg.xcl_profileLength = ROISize*0.8f;
1143     cfg.xcl_profileWidth = ROISize*0.2f;
1144     cfg.xcl_iterations = 1;
1145     cfg.testRun = true;
1146     cfg.Update();
1147     cfg.WriteToFile();
1148
1149     std::string file = SPrintf("%s\\lut%03d.jpg",zlutPath.c_str(),0);
1150     ImageData lut = ReadJPEGFile(file.c_str());
1151
1152     if(cfg.zlut_radialsteps != lut.w){
1153         output->outputString("ZLUT settings do not match LUT image sizes!",true);
1154         lut.free();
1155         return;
1156     }
1157
1158     int zplanes = lut.h;
1159     lut.free();
1160     int res = cfg.zlut_radialsteps;
1161
1162     float* zluts = new float[zplanes*res*beads.size()];
1163     ROIPosition* positions = new ROIPosition[beads.size()];
1164     for(int ii = 0; ii < beads.size(); ii++){
1165         std::string file = SPrintf("%s\\lut%03d.jpg",zlutPath.c_str(),ii);
1166         ImageData lut = ReadJPEGFile(file.c_str());
1167         memcpy(zluts+ii*res*zplanes,lut.data,res*zplanes*sizeof(float));
1168
1169         lut.free();
1170
1171         positions[ii].x = beads.at(ii).x - ROISize/2;
1172         positions[ii].y = beads.at(ii).y - ROISize/2;
1173     }
1174
1175     QueuedCPUTracker* qtrk = new QueuedCPUTracker(cfg);
1176     qtrk->SetRadialZLUT(zluts,beads.size(),zplanes);
1177     qtrk->FinalizeLUT();

```

```

1179 /*for(int ii = 0; ii < beads.size(); ii++){
1180     ImageData lut = ImageData::alloc(cfg.zlut_radialsteps,zplanes);
1181     memcpy(lut.data,qtrk->GetZLUTByIndex(ii),cfg.zlut_radialsteps*zplanes*sizeof(float));
1182     output->outputImage(lut,SPrintf("lut-%d,%d",beads.at(ii).x,beads.at(ii).y));
1183     lut.free();
1184 }*/
1185 //qtrk->ZLUTSelfTest();
1186
1187 qtrk->SetLocalizationMode(LT_QI | LT_LocalizeZ |
LT_NormalizeProfile);
1188
1189 ResultManagerConfig RMcfg;
1190 RMcfg.numBeads = beads.size();
1191 RMcfg.numFrameInfoColumns = 0;
1192 RMcfg.scaling = vector3f(1.0f,1.0f,1.0f);
1193 RMcfg.offset = vector3f(0.0f,0.0f,0.0f);
1194 RMcfg.writeInterval = 25;
1195 RMcfg.maxFramesInMemory = 100;
1196 RMcfg.binaryOutput = false;
1197
1198 std::vector<std::string> colnames;
1199 for(int ii = 0;ii<RMcfg.numFrameInfoColumns;ii++){
1200     colnames.push_back(SPrintf("%d",ii));
1201 }
1202
1203 ResultManager* RM = new ResultManager(
1204     SPrintf("%s\\RMOutput.txt",output->folder.c_str()).c_str(),
1205     SPrintf("%s\\RMFrameInfo.txt",output->folder.c_str()).c_str(),
1206     &RMcfg, colnames);
1207
1208 RM->SetTracker(qtrk);
1209
1210 int numFramesToTrack = NumJpgInDir(imagePath + "*");
1211 for(int ii = 0; ii < numFramesToTrack; ii++){
1212     std::string file = SPrintf("%s\img%05d.jpg",imagePath.c_str(),ii);
1213     ImageData img = ReadJPEGFile(file.c_str());
1214
1215     LocalizationJob job(ii, 0, 0, 0);
1216     int queued = qtrk->ScheduleFrame(img.data,sizeof(float)*img.w,img.w,img.h,positions,
beads.size(),QTrkFloat,&job);
1217     //output->outputString(SPrintf("Queueing frame %d. Current Queue size: %d. Added:
1218     %d.",ii,qtrk->GetQueueLength(),queued),true);
1219     ResultManager::FrameCounters cnt = RM->GetFrameCounters();
1220     output->outputString(SPrintf("Queueing frame %d. Current Queue size: %d.\n\t
captured: %d\n\tprocessed: %d\n\tlocalizations: %d\n\tLOST: %d\n\t",ii,qtrk->
GetQueueLength(),cnt.capturedFrames,cnt.
processedFrames,cnt.localizationsDone,cnt.
lostFrames),true);
1221
1222     img.free();
1223 }
1224
1225 while(qtrk->GetQueueLength() != 0);
1226 RM->Flush();
1227 delete RM;
1228
/* NO RESULT MANAGER
1229
int numFramesToTrack = NumJpgInDir(imagePath + "*");
for(int ii = 0; ii < numFramesToTrack; ii++){
    std::string file = SPrintf("%s\img%05d.jpg",imagePath.c_str(),ii);
    ImageData img = ReadJPEGFile(file.c_str());
1234
    LocalizationJob job(ii, 0, 0, 0);
    int queued = qtrk->ScheduleFrame(img.data,sizeof(float)*
img.w,img.w,img.h,positions,beads.size(),QTrkFloat,&job);
    output->outputString(SPrintf("Queueing frame %d. Current Queue size: %d. Added:
1236     %d.",ii,qtrk->GetQueueLength(),queued),true);
    img.free();
1237
    while(qtrk->GetResultCount() != 0) {
        LocalizationResult lr;
        qtrk->FetchResults(&lr,1);
        output->outputString(SPrintf("frame %d, bead %d: %f %f
1238     %f",lr.job.frame,lr.job.zlutIndex,lr.pos.x,lr.pos.y,lr.pos.z));
        //delete &(lr.job);
1239    }
1240
    while(qtrk->GetQueueLength() != 0) {
        if(qtrk->GetResultCount() != 0) {
            LocalizationResult lr;
            qtrk->FetchResults(&lr,1);
            output->outputString(SPrintf("frame %d, bead %d: %f %f
1241     %f",lr.job.frame,lr.job.zlutIndex,lr.pos.x,lr.pos.y,lr.pos.z));
1242    }
1243
    }
1244
}
1245
1246
1247
1248 while(qtrk->GetQueueLength() != 0) {
1249     if(qtrk->GetResultCount() != 0) {
1250         LocalizationResult lr;
1251         qtrk->FetchResults(&lr,1);
1252         output->outputString(SPrintf("frame %d, bead %d: %f %f
1253     %f",lr.job.frame,lr.job.zlutIndex,lr.pos.x,lr.pos.y,lr.pos.z));
1254    }
1255
}
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3079
3080
3081
3082
3083
3084
3085
3086
3087
3087
3088
3089
3089
3090
3091
3092
3093
3094
3095
3096
3097
3097
3098
3099
3099
3100
3101
3102
3103
3104
3105
3105
3106
3107
3108
3109
3109
3110
3111
3112
3113
3113
3114
3115
3116
3116
3117
3118
3119
3119
3120
3121
3122
3123
3123
3124
3125
3126
3126
3127
3128
3129
3129
3130
3131
3132
3133
3133
3134
3135
3136
3136
3137
3138
3139
3139
3140
3141
3142
3143
3143
3144
3145
3146
3146
3147
3148
3149
3149
3150
3151
3152
3153
3153
3154
3155
3156
3156
3157
3158
3159
3159
3160
3161
3162
3163
3163
3164
3165
3166
3166
3167
3168
3169
3169
3170
3171
3172
3173
3173
3174
3175
3176
3176
3177
3178
3179
3179
3180
3181
3182
3183
3183
3184
3185
3186
3186
3187
3188
3189
3189
3190
3191
3192
3193
3193
3194
3195
3196
3196
3197
3198
3199
3199
3200
3201
3202
3203
3203
3204
3205
3206
3206
3207
3208
3209
3209
3210
3211
3212
3213
3213
3214
3215
3216
3216
3217
3218
3219
3219
3220
3221
3222
3223
3223
3224
3225
3226
3226
3227
3228
3229
3229
3230
3231
3232
3233
3233
3234
3235
3236
3236
3237
3238
3239
3239
3240
3241
3242
3243
3243
3244
3245
3246
3246
3247
3248
3249
3249
3250
3251
3252
3253
3253
3254
3255
3256
3256
3257
3258
3259
3259
3260
3261
3262
3263
3263
3264
3265
3266
3266
3267
3268
3269
3269
3270
3271
3272
3273
3273
3274
3275
3276
3276
3277
3278
3279
3279
3280
3281
3282
3283
3283
3284
3285
3286
3286
3287
3288
3289
3289
3290
3291
3292
3293
3293
3294
3295
3296
3296
3297
3298
3299
3299
3300
3301
3302
3303
3303
3304
3305
3306
3306
3307
3308
3309
3309
3310
3311
3312
3313
3313
3314
3315
3316
3316
3317
3318
3319
3319
3320
3321
3322
3323
3323
3324
3325
3326
3326
3327
3328
3329
```

```

1253         }
1254     }
1255 */
1256 /*ImageData allLuts = ImageData::alloc(res,zplanes*beads.size());
1257 memcpy(allLuts.data,zluts,res*zplanes*beads.size()*sizeof(float));
1258 output->outputImage(allLuts,"allLuts");
1259 allLuts.free();*/
1260
1261 qtrk->ClearResults();
1262 delete qtrk;
1263 delete positions;
1264 delete[] zluts;
1265 }
```

11.2.2.20 void ScatterBiasArea (int *roi*, float *scan_width*, int *steps*, int *samples*, int *qi_it*, float *angstep*)

```

818 {
819     std::vector<float> u=linspace(roi/2-scan_width/2,roi/2+scan_width/2, steps);
820
821     QTrkComputedConfig cfg;
822     cfg.width=cfg.height=roi;
823     cfg.qi_angstep_factor = angstep;
824     cfg.qi_iterations = qi_it;
825     cfg.qi_angular_coverage = 0.7f;
826     cfg.qi_roi_coverage = 1;
827     cfg.qi_radial_coverage = 1.5f;
828     cfg.qi_minradius=0;
829     cfg.zlut_minradius=0;
830     cfg.zlut_angular_coverage = 0.7f;
831     cfg.zlut_roi_coverage = 1;
832     cfg.zlut_radial_coverage = 1.5f;
833     cfg.zlut_minradius = 0;
834     cfg.qi_minradius = 0;
835     cfg.com_bgcorrection = 0;
836     cfg.xcl_profileLength = roi*0.8f;
837     cfg.xcl_profileWidth = roi*0.2f;
838     cfg.xcl_iterations = 1;
839     cfg.Update();
840
841     ImageData lut,orglut = ReadLUTFile("10x.radialzlut#4");
842     vector3f ct(roi/2,roi/2,lut.h/2 + 0.123f);
843     float dx = scan_width/steps;
844
845     QueuedCPUTracker trk(cfg);
846     ResampleLUT(&trk, &orglut, orglut.h, &lut);
847     int maxval = 10000;
848
849     ImageData tmp=ImageData::alloc(roi,roi);
850     GenerateImageFromLUT(&tmp, &lut, 0, cfg.zlut_maxradius,
851     vector3f(roi/2,roi/2,lut.h/2));
852     ApplyPoissonNoise(tmp, maxval);
853
854     std::string fn = SPrintf( "sb_area_roi%d_scan%d_steps%d_qit%d_N%d", roi, (int)scan_width, steps,
855     qi_it, samples);
856     WriteJPEGFile( (fn + ".jpg").c_str(), tmp);
857     tmp.free();
858
859     fn += ".txt";
860     for (int y=0;y<steps;y++) {
861         for (int x=0;x<steps;x++)
862         {
863             vector3f cpos( (x+0.5f-steps/2) * dx, (y+0.5f-steps/2) * dx, 0 );
864
865             cfg.qi_iterations = qi_it;
866             auto r= AccBiasTest(orglut, &trk, samples, cpos+ct,
867             vector3f(), 0, maxval, qi_it < 0 ? LT_XCor1D : 0);
868
869             float row[] = { r.acc.x, r.acc.y, r.acc.z, r.bias.x, r.bias.y, r.bias.z, r.crlb.x, r.crlb.z,
870             samples };
871             WriteArrayAsCSVRow(fn.c_str(), row, 9, x+y>0);
872             dbgprintf("X=%d,Y=%d\n", x,y);
873         }
874     }
875     orglut.free();
876     lut.free();
877 }
```

11.2.2.21 void SelectTests (const char * image, int OutputMode)

```

1437 {
1438     outputter* output = new outputter(OutputMode);
1439     int testNum = 1;
1440     int ROISize = 120;
1441
1442     PrintMenu(output);
1443     char inChar;
1444     do
1445     {
1446         output->outputString("Select test or ? for menu:",true);
1447         std::string imagesPath, LUTPath;
1448         std::cin >> inChar;
1449         switch(inChar)
1450         {
1451             case '0':
1452                 break;
1453             case '1':
1454                 RunTest(ROIDis,image,output,ROISize);
1455                 output->outputString("Test done!",true);
1456                 break;
1457             case '2':
1458                 RunTest(Inter,image,output,ROISize);
1459                 output->outputString("Test done!",true);
1460                 break;
1461             case '3':
1462                 RunTest(Skew,image,output,ROISize);
1463                 output->outputString("Test done!",true);
1464                 break;
1465             case '4':
1466                 RunTest(Backg,image,output,ROISize);
1467                 output->outputString("Test done!",true);
1468                 break;
1469             case '5':
1470                 //BuildZLUT("L:\\\\BN\\\\ND\\\\Shared\\\\Jordi\\\\TestMovie150507_2\\\\images\\\\jpg\\\\Zstack\\\\",output);
1471                 BuildZLUT("D:\\\\TestImages\\\\TestMovie150507_2\\\\images\\\\jpg\\\\Zstack\\\\",output);
1472                 output->outputString("ZLUTs Built",true);
1473                 break;
1474             case '6':
1475                 /*while(!DirExists(imagesPath) && NumJpgInDir(imagesPath) > 0)){
1476                     output->outputString("Enter image folder (with beadlist):",true);
1477                     std::cin >> imagesPath;
1478                     output->outputString(SPrintf("%d jpgs in folder",NumJpgInDir(imagesPath)),true);
1479                 }
1480                 while(!DirExists(LUTPath) && NumJpgInDir(LUTPath) > 0)){
1481                     output->outputString("Enter LUT folder:",true);
1482                     std::cin >> LUTPath;
1483                     output->outputString(SPrintf("%d jpgs in folder",NumJpgInDir(LUTPath)),true);
1484                 }
1485                 RunZTrace(imagesPath,LUTPath,output);*/
1486             /*
1487                 RunZTrace("L:\\\\BN\\\\ND\\\\Shared\\\\Jordi\\\\20150804_TestMovie\\\\images\\\\jordi_test\\\\jpg\\\\","L:\\\\BN\\\\ND\\\\Shared\\\\Jordi\\\\20150804_TestMovie\\\\images\\\\jordi_test\\\\jpg\\\\");
1488                 RunZTrace("D:\\\\TestImages\\\\20150804_TestMovie\\\\images\\\\jordi_test\\\\jpg\\\\","D:\\\\TestImages\\\\20150804_TestMovie\\\\images\\\\jordi_test\\\\jpg\\\\");
1489                 RunZTrace("C:\\\\TestImages\\\\TestMovie150507_2\\\\images\\\\jpg\\\\Zstack\\\\","C:\\\\TestImages\\\\TestMovie150507_2\\\\images\\\\jpg\\\\Zstack\\\\");
1490             */
1491             case 'm':
1492                 ManTest();
1493                 break;
1494             case 'R':
1495             case 'r':
1496                 output->outputString(SPrintf("Enter new ROI size (currently %d)",ROISize
1497 ),true);
1498                 std::cin >> ROISize;
1499                 break;
1500             default:
1501                 output->outputString("Wrong input",true);
1502             case '?':
1503                 PrintMenu(output);
1504                 break;
1505             }
1506         } while(inChar != '0');
1507     delete output;
1508 }
```

11.2.2.22 void SimpleTest ()

704 {

```

705     QTrkSettings cfg;
706     cfg.qi_minradius=0;
707     cfg.zlut_minradius = 0;
708     cfg.width = cfg.height = 30;
709     auto locModeQI = (LocMode_t)(LT_QI | LT_NormalizeProfile |
710     LT_LocalizeZ);
711     auto results = RunTracker<QueuedCPUTracker> ("lut000.jpg", &cfg, false, "qi", locModeQI, 1000, 10000/
712     255 );
713     results.computeStats();
714     dbgprintf("X= %f. stdev: %f\tZ=%f, stdev: %f\n",
715         results.meanErr.x, results.stdev.x, results.meanErr.z, results.stdev.z);
716 }
```

11.2.2.23 float SkewParam (ImageData img)

```

901     {
902     int size = img.w * 2 + img.h * 2 - 4;
903     float* outeredge = new float[size];
904     GetOuterEdges(outeredge,size,img);
905     float* max = std::max_element(img.data,img.data+(img.w*img.h));
906     float* min = std::min_element(img.data,img.data+(img.w*img.h));
907     float* outer_max = std::max_element(outeredge,outeredge+size);
908     float* outer_min = std::min_element(outeredge,outeredge+size);
909
910     float out = (*outer_max-*outer_min)/(*max-*min);
911     delete[] outeredge;
912     return out;
913 }
```

11.2.2.24 void SmallImageTest()

```

150 {
151     CPUTracker *tracker = new CPUTracker(50,50, 16);
152
153     GenerateTestImage(ImageData(tracker->srcImage, tracker->
154     GetWidth(), tracker->GetHeight()), tracker->width/2,tracker->
155     height/2, 9, 0.0f);
156     FloatToJPEGFile("smallimg.jpg", tracker->srcImage, tracker->
157     width, tracker->height);
158
159     vector2f com = tracker->ComputeMeanAndCOM(0);
160     dbgout(SPrintf("COM: %f,%f\n", com.x, com.y));
161
162     vector2f initial(25,25);
163     bool boundaryHit = false;
164     vector2f xcor = tracker->ComputeXCorInterpolated(initial, 2, 16,
165     boundaryHit);
166     dbgout(SPrintf("XCor: %f,%f\n", xcor.x, xcor.y));
167     //assert(fabsf(xcor.x-15.0f) < 1e-6 && fabsf(xcor.y-15.0f) < 1e-6);
168
169     int I=4;
170     vector2f pos = initial;
171     for (int i=0;i<I;i++) {
172         bool bhit;
173         vector2f np = tracker->ComputeQI(pos, 1, 32, 4, 1, 1, 16, bhit);
174         dbgprintf("qi[%d]. New=%4f, %4f;\tOld=%4f, %4f\n", i, np.x, np.
175         y, pos.x, pos.y);
176     }
177
178     FloatToJPEGFile("debugimg.jpg", tracker->GetDebugImage(), tracker->
179     width, tracker->height);
180     delete tracker;
181 }
```

11.2.2.25 void SpeedTest()

```

37 {
38 #ifdef _DEBUG
39     int N = 20;
40 #else
41     int N = 1000;
42 #endif
43     int qi_iterations = 7;
44     int xcor_iterations = 7;
45     CPUTracker* tracker = new CPUTracker(150,150, 128);
46
47     int radialSteps = 64, zplanes = 120;
48     float zmin = 2, zmax = 8;
49     float zradius = tracker->xcorw/2;
50
51     float* zlut = new float[radialSteps*zplanes];
52     for (int x=0;x<zplanes;x++) {
53         vector2f center (tracker->GetWidth()/2, tracker->
54             GetHeight()/2);
55         float s = zmin + (zmax-zmin) * x/(float)(zplanes-1);
56         GenerateTestImage(ImageData(tracker->srcImage, tracker->
57             GetWidth(), tracker->GetHeight()), center.x, center.y, s, 0.0f);
58         tracker->mean = 0.0f;
59         tracker->ComputeRadialProfile(&zlut[x*radialSteps], radialSteps, 64, 1, zradius
60             , center, false);
61     }
62     tracker->SetRadialZLUT(zlut, zplanes, radialSteps, 1,1, zradius, true, true);
63     delete[] zlut;
64
65 // Speed test
66 vector2f comdist, xcordist, qidist;
67 float zdist=0.0f;
68 double zerrsum=0.0f;
69 double tcom = 0.0, tgen=0.0, tz = 0.0, tqi=0.0, txcor=0.0;
70 for (int k=0;k<N;k++)
71 {
72     double t0 = GetPreciseTime();
73     float xp = tracker->GetWidth()/2+(rand_uniform<float>() - 0.5) * 5;
74     float yp = tracker->GetHeight()/2+(rand_uniform<float>() - 0.5) * 5;
75     float z = zmin + 0.1f + (zmax-zmin-0.2f) * rand_uniform<float>();
76
77     GenerateTestImage(ImageData(tracker->srcImage, tracker->
78         GetWidth(), tracker->GetHeight()), xp, yp, z, 0);
79
80     double t1 = GetPreciseTime();
81     vector2f com = tracker->ComputeMeanAndCOM();
82     vector2f initial(com.x, com.y);
83     double t2 = GetPreciseTime();
84     bool boundaryHit = false;
85     vector2f xcor = tracker->ComputeXCorInterpolated(initial,
86         xcor_iterations, 16, boundaryHit);
87     if (boundaryHit)
88         dbgprintf("xcor boundaryhit!!\n");
89
90     comdist.x += fabsf(com.x - xp);
91     comdist.y += fabsf(com.y - yp);
92
93     xcordist.x +=fabsf(xcordist.x - xp);
94     xcordist.y +=fabsf(xcordist.y - yp);
95     double t3 = GetPreciseTime();
96     boundaryHit = false;
97     vector2f qi = tracker->ComputeQI(initial, qi_iterations, 64, 16,
98         ANGSTEPF, 5,50, boundaryHit);
99     qidist.x += fabsf(qi.x - xp);
100    qidist.y += fabsf(qi.y - yp);
101    double t4 = GetPreciseTime();
102    if (boundaryHit)
103        dbgprintf("qi boundaryhit!!\n");
104
105    boundaryHit = false;
106    float est_z = zmin + (zmax-zmin)*tracker->ComputeZ(qi, 64, 0, &boundaryHit, 0) / (zplanes-1
107    );
108    zdist += fabsf(est_z-z);
109    zerrsum += est_z-z;
110
111    if (boundaryHit)
112        dbgprintf("computeZ boundaryhit!!\n");
113    double t5 = GetPreciseTime();
114    // dbgout(SPrintf("xpos:%f, COM err: %f, XCor err: %f\n", xp, com.x-xp, xcor.x-xp));
115    if (k>0) { // skip first initialization round
116        tgen+=t1-t0;
117        tcom+=t2-t1;
118        txcor+=t3-t2;
119        tqi+=t4-t3;
120        tz+=t5-t4;
121    }
122}

```

```

114         }
115     }
116
117     int Nns = N-1;
118     dbgprintf("Image gen. (img/s): %f\nCenter-of-Mass speed (img/s): %f\n", Nns/tgen, Nns/tcom);
119     dbgprintf("XCor estimation (img*it/s): %f\n", (Nns*xcor_iterations)/txcor);
120     dbgprintf("COM+XCor(%d) (img/s): %f\n", xcor_iterations, Nns/(tcom+txcor));
121     dbgprintf("Z estimation (img/s): %f\n", Nns/tz);
122     dbgprintf("QI speed: %f (img*it/s)\n", (Nns*qi_iterations)/tqi);
123     dbgprintf("Average dist: COM x: %f, y: %f\n", comdist.x/N, comdist.y/N);
124     dbgprintf("Average dist: Cross-correlation x: %f, y: %f\n", xcordist.x/N, xcordist.y/N);
125     dbgprintf("Average dist: QI x: %f, y: %f\n", qidist.x/N, qidist.y/N);
126     dbgprintf("Average dist: Z: %f. Mean error:%f\n", zdist/N, zerrsum/N);
127
128     delete tracker;
129 }
```

11.2.2.26 void TestBackground (std::vector< BeadPos > beads, ImageData orilimg, outputter * output, int ROISize)

```

977 {
978     std::string out;
979
980     out = "Beads-Background";
981     output->newFile(out);
982
983     for(uint ii = 0; ii < beads.size(); ii++){
984         int x = beads.at(ii).x - ROISize/2;
985         int y = beads.at(ii).y - ROISize/2;
986         ImageData img = CropImage(orilimg,x,y,ROISize,ROISize);
987         output->outputImage(img,SPrintf("%d,%d-Crop",beads.at(ii).x, beads.at(ii).y));
988         output->outputString(SPrintf("%d,%d-Crop",beads.at(ii).x, beads.at(ii).y));
989         output->outputString(SPrintf("median: %f",
990             BackgroundMedian(img)));
990         output->outputString(SPrintf("sigma: %f",
991             BackgroundStdDev(img)));
992         output->outputString(SPrintf("rms: %f",BackgroundRMS(img)));
993         output->outputString("");
994     }
995 }
```

11.2.2.27 void TestBias ()

```

374 {
375     ImageData lut = ReadLUTFile("lut000.jpg");
376
377     ImageData img = ImageData::alloc(80,80);
378     CPUTracker trk(img.w,img.h);
379
380     float o = 2.03f;
381     vector3f pos (img.w/2+o, img.h/2+o, lut.h/2);
382     NormalizeZLUT(lut.data, 1, lut.h, lut.w);
383     lut.normalize();
384
385     srand(time(0));
386     for (int i=0;i<10;i++) {
387         GenerateImageFromLUT(&img, &lut, 0, img.w/2, pos, true);
388         ApplyPoissonNoise(img, 11050);
389         float stdev = StdDeviation(img.data, img.data + img.
390 w);
390         dbgprintf("Noise level std: %f\n",stdev);
391     }
392
393     WriteJPEGFile("TestBias-smp.jpg", img);
394     trk.setImageFloat(img.data);
395
396     vector2f com = trk.ComputeMeanAndCOM();
397     dbgprintf("COM: x:%f, y:%f\n", com.x,com.y);
398     bool bhit;
399     auto rw = ComputeRadialBinWindow(img.w);
400     vector2f qi = trk.ComputeQI(com, 3, img.w, 3*img.w/4, 1, 0, img.w/2, bhit, &rw[0]);
401     dbgprintf("QI: x: %f, y:%f\n", qi.x,qi.y);
402
403     trk.setRadialZLUT(lut.data, lut.h, lut.w, 1, 0, img.w/2, true, false);
404     float z = trk.ComputeZ(qi,3*img.w, 0, 0, 0, 0, false);
```

```

405     float znorm = trk.ComputeZ(qi, 3*img.w, 0, 0, 0, 0, true);
406     dbgprintf("Z: %f, ZNorm: %f, true:%f\n", z, znorm, pos.z);
408
409     img.free();
410     lut.free();
411 }

```

11.2.2.28 void TestBoundCheck()

```

206 {
207     CPUTracker *tracker = new CPUTracker(32,32, 16);
208     bool boundaryHit;
209
210     for (int i=0;i<10;i++) {
211         float xp = tracker->GetWidth()/2+(rand_uniform<float>() - 0.5) * 20;
212         float yp = tracker->GetHeight()/2+(rand_uniform<float>() - 0.5) * 20;
213
214         GenerateTestImage(ImageData(tracker->srcImage, tracker->
215             GetWidth(), tracker->GetHeight()), xp, yp, 1, 0.0f);
216
217         vector2f com = tracker->ComputeMeanAndCOM();
218         dbgout(SPrintf("COM: %f,%f\n", com.x-xp, com.y-yp));
219
220         vector2f initial = com;
221         boundaryHit=false;
222         vector2f xcor = tracker->ComputeXCorInterpolated(initial, 3, 16,
223             boundaryHit);
224         dbgprintf("XCor: %f,%f. Err: %d\n", xcor.x-xp, xcor.y-yp, boundaryHit);
225
226         boundaryHit=false;
227         vector2f qi = tracker->ComputeQI(initial, 3, 64, 32,
228             ANGSTEPF, 1, 10, boundaryHit);
229         dbgprintf("QI: %f,%f. Err: %d\n", qi.x-xp, qi.y-yp, boundaryHit);
230     }
231
232     delete tracker;
233 }

```

11.2.2.29 static void TestBSplineMax (float maxpos) [static]

```

718 {
719     const int N=100;
720     float v[N];
721
722     for (int i=0;i<N;i++)
723         v[i] = -sqrt(1+(i-maxpos)*(i-maxpos));
724     float max = ComputeSplineFitMaxPos(v, N);
725     dbgprintf("Max: %f, true: %f\n", max, maxpos);
726 }

```

11.2.2.30 void TestFourierLUT()

```

613 {
614     QTrkSettings cfg;
615     cfg.width = cfg.height = 60;
616     cfg.zlut_minradius=3;
617     cfg.zlut_roi_coverage=1;
618
619 //    auto locMode = (LocMode_t)(LT_ZLUTAlign | LT_NormalizeProfile | LT_LocalizeZ);
620 //    auto resultsCOM = RunTracker<QueuedCPUTracker> ("lut000.jpg", &cfg, false, "com-zlутalign", locMode,
621 //        100 );
621
622     const float NF=28;
623     float zpos=10;
624
625     auto locMode = (LocMode_t)(LT_QI | LT_FourierLUT |
626     LT_NormalizeProfile | LT_LocalizeZ);
626     auto resultsZA = RunTracker<QueuedCPUTracker> ("lut000.jpg", &cfg, false, "qi-fourierlut", locMode,
627     200, NF,zpos);
627

```

```

628     auto locModeQI = (LocMode_t)(LT_QI | LT_NormalizeProfile |
629     LT_LocalizeZ);
630     auto resultsQI = RunTracker<QueuedCPUTracker> ("lut000.jpg", &cfg, false, "qi", locModeQI, 200, NF,
631     zpos);
632     resultsZA.computeStats();
633     resultsQI.computeStats();
634     dbgprintf("FourierLUT: X= %f. stdev: %f\|tZ=%f, stdev: %f\n", resultsZA.meanErr.x, resultsZA.
635     stdev.x, resultsZA.meanErr.z, resultsZA.stdev.z);
636     dbgprintf("Only QI: X= %f. stdev: %f\|tZ=%f, stdev: %f\n", resultsQI.meanErr.x, resultsQI.
637     stdev.x, resultsQI.meanErr.z, resultsQI.stdev.z);
638 }
```

11.2.2.31 void TestFourierLUTOnDataset()

```

639 {
640     const char*basepath= "D:/jcrossen1/datasets/RefBeads 2013-09-02/2013-09-02/";
641
642     process_bead_dir(SPrintf("%s/%s", basepath, "tmp_001").c_str(), 80, [&] (
643         ImageData *img, int bead, int frame) {
644
645             }, 1000);
646 }
```

11.2.2.32 void TestInterference (std::vector< BeadPos > beads, ImageData oriImg, outputter * output, int ROISize, vector2f displacement = vector2f (60, 0))

```

936 {
937     ImageData added = AddImages(oriImg, oriImg, displacement);
938     //output->outputImage(added, "Added");
939
940     for(uint ii = 0; ii < beads.size(); ii++) {
941         output->newFile(SPrintf("%d,%d-Inter", beads.at(ii).x, beads.at(ii).y));
942
943         int x = beads.at(ii).x - ROISize/2;
944         int y = beads.at(ii).y - ROISize/2;
945
946         ImageData img = CropImage(added, x, y, ROISize, ROISize);
947         output->outputImage(img, SPrintf("%d,%d-Inter", beads.at(ii).x, beads.at(ii).y));
948
949         RunCOMAndQI(img, output);
950         img.free();
951     }
952
953     added.free();
954 }
```

11.2.2.33 void TestQuadrantAlign ()

```

673 {
674     QTrkSettings cfg;
675     cfg.width = cfg.height = 60;
676     cfg.numThreads=1;
677
678 //    auto locMode = (LocMode_t)(LT_ZLUTAlign | LT_NormalizeProfile | LT_LocalizeZ);
679 //    auto resultsCOM = RunTracker<QueuedCPUTracker> ("lut000.jpg", &cfg, false, "com-zlутalign", locMode,
680 //    100 );
681     const float NF=10;
682
683 #ifdef _DEBUG
684     int N=1;
685     cfg.numThreads=1;
686 #else
687     int N=2000;
688 #endif
689     auto locModeQI = (LocMode_t)(LT_QI | LT_NormalizeProfile |
```

```

    LT_LocalizeZ);
691   auto resultsQI = RunTracker<QueuedCPUTracker> ("lut000.jpg", &cfg, false, "qa", locModeQI, N, NF);
692
693   auto locMode = (LocMode_t)(LT_QI | LT_ZLUTAlign |
694     LT_NormalizeProfile | LT_LocalizeZ);
695   auto resultsZA = RunTracker<QueuedCPUTracker> ("lut000.jpg", &cfg, false, "qa-qalign", locMode, N, NF );
696
697   ;
698
699   resultsZA.computeStats();
700   resultsQI.computeStats();
701
702
703   dbgprintf("QuadrantAlign: X= %f. stdev: %f\|tZ=%f, stdev: %f\n", resultsZA.meanErr.x,
704   resultsZA.stdev.x, resultsZA.meanErr.z, resultsZA.stdev.z);
705   dbgprintf("Only QI: X= %f. stdev: %f\|tZ=%f, stdev: %f\n", resultsQI.meanErr.x, resultsQI.
706   stdev.x, resultsQI.meanErr.z, resultsQI.stdev.z);
707 }

```

11.2.2.34 void TestROIDisplacement (std::vector< BeadPos > beads, ImageData oriImg, outputter * output, int ROISize, int maxdisplacement = 0)

```

916 {
917   for(uint ii = 0; ii < beads.size(); ii++){
918
919     output->newFile(SPrintf("%d,%d-ROI-%d",beads.at(ii).x,beads.at(ii).y,ROISize));
920
921     for(int x_i = -maxdisplacement; x_i <= maxdisplacement; x_i++){
922       for(int y_i = -maxdisplacement; y_i <= maxdisplacement; y_i++){
923         int x = beads.at(ii).x + x_i - ROISize/2;
924         int y = beads.at(ii).y + y_i - ROISize/2;
925         ImageData img = CropImage(oriImg,x,y,ROISize,ROISize);
926         output->outputImage(img,Sprintf("%d,%d-Crop",beads.at(ii).x + x_i, beads.
927         at(ii).y + y_i));
928         output->outputString(SPrintf("%d,%d - ROI (%d,%d) -> (%d,%d)",x_i,y_i,x,
929         y,x+ROISize,y+ROISize));
930         RunCOMAndQI(img,output);
931         img.free();
932       }
933     }
934   }

```

11.2.2.35 void TestSkew (std::vector< BeadPos > beads, ImageData oriImg, outputter * output, int ROISize)

```

957 {
958   for(uint ii = 0; ii < beads.size(); ii++){
959     output->newFile(SPrintf("%d,%d-Skew",beads.at(ii).x,beads.at(ii).y));
960
961     int x = beads.at(ii).x - ROISize/2;
962     int y = beads.at(ii).y - ROISize/2;
963
964     ImageData img = CropImage(oriImg,x,y,ROISize,ROISize);
965     ImageData skewImg = SkewImage(img,20);
966     output->outputImage(skewImg,Sprintf("%d,%d-Skew",beads.at(ii).x,beads.at(ii).y));
967     float imgSkew = SkewParam(img);
968     float skewImgSkew = SkewParam(skewImg);
969     output->outputString(SPrintf("%f %f",imgSkew,skewImgSkew));
970     RunCOMAndQI(skewImg,output);
971     img.free();
972     skewImg.free();
973   }
974 }

```

11.2.2.36 void TestZLUTAlign ()

```

650 {
651   QTrkSettings cfg;
652   cfg.width = cfg.height = 60;
653
654 //  auto locMode = (LocMode_t)(LT_ZLUTAlign | LT_NormalizeProfile | LT_LocalizeZ);
655 //  auto resultsCOM = RunTracker<QueuedCPUTracker> ("lut000.jpg", &cfg, false, "com-zlутalign", locMode,
656 //  100 );

```

```

656
657     const float NF=28;
658
659     auto locModeQI = (LocMode_t)(LT_QI | LT_NormalizeProfile |
660     LT_LocalizeZ);
660     auto resultsQI = RunTracker<QueuedCPUTracker> ("lut000.jpg", &cfg, false, "qi", locModeQI, 200, NF);
661
662     auto locMode = (LocMode_t)(LT_QI | LT_ZLUTAlign |
663     LT_NormalizeProfile | LT_LocalizeZ);
663     auto resultsZA = RunTracker<QueuedCPUTracker> ("lut000.jpg", &cfg, false, "qi-zlутalign", locMode, 200,
664     NF );
664
665     resultsZA.computeStats();
666     resultsQI.computeStats();
667
668     dbgprintf("ZLUTAlign: X= %f. stdev: %f\ tZ=%f, stdev: %f\n", resultsZA.meanErr.x, resultsZA.
669     stdev.x, resultsZA.meanErr.z, resultsZA.stdev.z);
670     dbgprintf("Only QI: X= %f. stdev: %f\ tZ=%f, stdev: %f\n", resultsQI.meanErr.x, resultsQI.
671     stdev.x, resultsQI.meanErr.z, resultsQI.stdev.z);
670 }

```

11.2.2.37 void TestZRange (const char * name, const char * lutfile, int extraFlags, int clean_lut, RWeightMode weightMode = RWNone, bool biasMap = false, bool biasCorrect = false)

```

453 {
454     ImageData lut = ReadLUTFile(lutfile);
455     vector3f delta(0.001f, 0.001f, 0.001f);
456
457     if(PathSeparator(lutfile).extension != "jpg"){
458         WriteJPEGFile(SPrintf("%s-lut.jpg",name).c_str(), lut);
459     }
460
461     if (clean_lut) {
462         BenchmarkLUT::CleanupLUT(lut);
463         WriteJPEGFile( std::string(lutfile).substr(0, strlen(lutfile)-4).append("_bmlut.jpg") .
464         c_str(), lut );
465     }
466
467     QTrkComputedConfig settings;
468     settings.qi_iterations = 2;
469     settings.zlut_minradius = 1;
470     settings.qi_minradius = 1;
471     settings.width = settings.height = 100;
472     settings.Update();
473
474     float maxVal=10000;
475     std::vector<float> stdv;
476     dbgprintf("High-res LUT range...\n");
477     SampleFisherMatrix fm( maxVal );
478
479     QueuedCPUTracker trk(settings);
480     ImageData rescaledLUT;
481     ResampleLUT(&trk, &lut, lut.h, &rescaledLUT);
482
483     if (biasCorrect) {
484         CImageData result;
485         trk.ComputeZBiasCorrection(lut.h*10, &result, 4, true);
486
487         WriteImageAsCSV(SPrintf("%s-biasc.txt", name).c_str(), result.
488         data, result.w, result.h);
489     }
490
491     int f = 0;
492     if (weightMode == RWDerivative)
493         f |= LT_LocalizeZWeighted;
494     else if(weightMode == RWRadial) {
495         std::vector<float> w(settings.zlut_radialsteps);
496         for (int i=0;i<settings.zlut_radialsteps;i++)
497             w[i]= settings.zlut_minradius + i/(float)settings.
498             zlut_radialsteps*settings.zlut_maxradius;
499         trk.SetRadialWeights(&w[0]);
500     }
501     else if (weightMode == RWStetson)
502         trk.SetRadialWeights( ComputeRadialBinWindow(settings.
503             zlut_radialsteps) );
504
505     trk.SetLocalizationMode(LT_QI|LT_LocalizeZ|
506     LT_NormalizeProfile|extraFlags|f);
507
508     uint nstep= InDebugMode ? 20 : 1000;
509     uint smpPerStep = InDebugMode ? 2 : 200;

```

```

505     if (biasMap) {
506         smpPerStep=1;
507         nstep=InDebugMode? 200 : 2000;
508     }
509
510     std::vector<vector3f> truepos, positions,crlb;
511     std::vector<float> stdevz;
512     for (uint i=0;i<nstep;i++)
513     {
514         float z = 1 + i / (float)nstep * (rescaledLUT.h-2);
515         vector3f pos = vector3f(settings.width/2, settings.
516         height/2, z);
517         truepos.push_back(pos);
518         Matrix3X3 invFisherLUT = fm.Compute(pos, delta, rescaledLUT, settings.
519         width, settings.height, settings.zlut_minradius, settings.
520         zlut_maxradius).Inverse();
521         crlb.push_back(sqrt(invFisherLUT.diag()));
522         ImageData img=ImageData::alloc(settings.width,settings.
523         height);
524         for (uint j=0;j<smpPerStep; j++) {
525             vector3f rndvec(rand_uniform<float>(), rand_uniform<float>(), rand_uniform<float>());
526             if (biasMap) rndvec=vector3f();
527             vector3f rndpos = pos + vector3f(1,1,0.1) * (rndvec-0.5f); // 0.1 plane is
528             still a lot larger than the 0.02 typical accuracy
529             GenerateImageFromLUT(&img, &rescaledLUT, settings.
530             zlut_minradius, settings.zlut_maxradius, rndpos, true);
531             img.normalize();
532             if (!biasMap) ApplyPoissonNoise(img, maxVal);
533             LocalizationJob job(positions.size(), 0, 0, 0);
534             trk.ScheduleImageData(&img, &job);
535             positions.push_back(rndpos);
536             if(j==0 && InDebugMode) {
537                 WriteJPEGFile(Sprintf("%s-sampleimg.jpg",name).c_str(), img);
538             }
539             dbgprintf("[%d] z=%f Min std deviation: X=%f, Y=%f, Z=%f.\n", i, z, crlb[i].x,crlb[i].y,
540             crlb[i].z);
541             img.free();
542         }
543         WaitForFinish(&trk, positions.size());
544         std::vector<vector3f> trkmean(nstep), trkstd(nstep);
545         std::vector<vector3f> resultpos(nstep*smpPerStep);
546         for (uint i=0;i<positions.size();i++) {
547             LocalizationResult lr;
548             trk.FetchResults(&lr, 1);
549             resultpos[lr.job.frame]=lr.pos;
550         }
551         for (uint i=0;i<nstep;i++) {
552             for (uint j=0;j<smpPerStep;j++) {
553                 vector3f err=resultpos[i*smpPerStep+j]-positions[i*smpPerStep+j];
554                 trkmean[i]+=err;
555             }
556             trkmean[i]/=smpPerStep;
557             vector3f variance;
558             for (uint j=0;j<smpPerStep;j++) {
559                 vector3f r = resultpos[i*smpPerStep+j];
560                 vector3f t = positions[i*smpPerStep+j];
561                 vector3f err=r-t;
562                 err -= trkmean[i];
563                 variance += err*err;
564             }
565             if (InDebugMode) {
566                 dbgprintf("Result: x=%f,y=%f,z=%f. True: x=%f,y=%f,z=%f\n", r.
567                 x,r.y,r.z,t.x,t.y,t.z);
568             }
569             if (biasMap) trkstd[i]=vector3f();
570             else trkstd[i] = sqrt(variance / (smpPerStep-1));
571         }
572         vector3f mean_std;
573         std::vector<float> output;
574         for(uint i=0;i<nstep;i++) {
575             dbgprintf("trkstd[%d]:%f crlb=%f bias=%f true=%f\n", i, trkstd[i].z, crlb[i].z, trkmean[i].
576             z, truepos[i].z);
577             output.push_back(truepos[i].z);
578             output.push_back(trkmean[i].x);
579             output.push_back(trkstd[i].x);
580             output.push_back(trkmean[i].z);
581             output.push_back(trkstd[i].z);
582             output.push_back(crlb[i].x);
583             output.push_back(crlb[i].z);
584         }
585         mean_std += trkstd[i];
586     }

```

```

583     dbgprintf("mean z err: %f\n", (mean_std/nstep).z);
584     WriteImageAsCSV( SPrintf("%s_%d_flags%d_cl%d.txt",name, weightMode, extraFlags,
585     clean_lut).c_str(), &output[0], 7, output.size()/7);
586     lut.free();
587     rescaledLUT.free();
588 }

```

11.2.2.38 void TestZRangeBias (const char * name, const char * lutfile, bool normProf)

```

414 {
415     ImageData lut = ReadLUTFile(lutfile);
416
417     QTrkComputedConfig settings;
418     settings.width=settings.height=100;
419     settings.Update();
420     ImageData img=ImageData::alloc(settings.width,settings.
421     height);
422     CPUTracker trk(settings.width,settings.height);
423     NormalizeZLUT(lut.data, 1, lut.h, lut.w);
424     trk.SetRadialZLUT(lut.data, lut.h, lut.w, 1, 1, settings.qi_maxradius, true, false);
425     float* prof= ALLOCA_ARRAY(float,lut.w);
426
427     int N=1000;
428     std::vector< vector2f> results (N);
429     for (int i=0;i<N;i++) {
430         vector3f pos (settings.width/2, settings.height/2, i/(float)N*lut.
431         h);
432         GenerateImageFromLUT(&img, &lut, 1, settings.
433         qi_maxradius, pos);
434         if(InDebugMode&&i==0){
435             WriteJPEGFile(SPrintf("%s-smp.jpg",name).c_str(),img);
436         }
437         trk.SetImageFloat(img.data);
438         //trk.ComputeRadialProfile(prof,lut.w,settings.zlut_angularsteps, settings.zlut_minradius,
439         settings.zlut_maxradius, pos.xy(), false);
440         //float z=trk.LUTProfileCompare(prof, 0, 0, normProf ? CPUTracker::LUTProfMaxSplineFit :
441         CPUTracker::LUTProfMaxQuadraticFit); result: splines no go!
442         float z = trk.ComputeZ(pos.xy(), settings.zlut_angularsteps, 0, 0, 0, 0,true);
443         results[i].x = pos.z;
444         results[i].y = z-pos.z;
445     }
446     lut.free();
447     img.free();
448 }

```

11.2.2.39 void WriteRadialProf (const char * file, ImageData & d)

```

351 {
352     CPUTracker trk(d.w,d.h);
353     trk.SetImageFloat(d.data);
354     vector2f com = trk.ComputeMeanAndCOM();
355     bool bhit;
356     vector2f qipos = trk.ComputeQI(com, 4, 64, 64,ANGSTEPF, 5, 50, bhit);
357
358     const int radialsteps=64;
359     float radprof[radialsteps];
360     trk.ComputeRadialProfile(radprof, radialsteps, 128, 5, 40, qipos, false);
361
362     WriteImageAsCSV(file, radprof, radialsteps, 1);
363 }

```

11.2.3 Variable Documentation

11.2.3.1 const float ANGSTEPF = 1.5f

11.2.3.2 const bool InDebugMode

Initial value:

```
=
```

```
false
```

11.3 cputrack-test/SharedTests.h File Reference

```
#include "random_distr.h"
#include <direct.h>
#include "FisherMatrix.h"
```

Classes

- struct [RunTrackerResults](#)
- struct [SpeedAccResult](#)

Functions

- template<typename T >
void [ResampleLUT](#) (T *qtrk, [ImageData](#) *lut, int zplanes, [ImageData](#) *newlut, const char *jpgfile=0, uint buildLUTFlags=0)
- static std::vector< [vector3f](#) > [FetchResults](#) ([QueuedTracker](#) *trk)
- static void [RandomFill](#) (float *d, int size, float mean, float std_deviations)
- template<typename TrackerType >
void [TestCMOSNoiseInfluence](#) (const char *lutfile)
- static void [EnableGainCorrection](#) ([QueuedTracker](#) *qtrk)
- template<typename TrackerType >
std::vector< [vector3f](#) > [Gauss2DTest](#) (int NumImages=10, int JobsPerImg=1000, bool useGC=false)
- static double [WaitForFinish](#) ([QueuedTracker](#) *qtrk, int N)
- static void [MeanStDevError](#) (const std::vector< [vector3f](#) > &truepos, const std::vector< [vector3f](#) > &v, [vector3f](#) &meanErr, [vector3f](#) &stdev)
- template<typename TrkType >
[RunTrackerResults](#) [RunTracker](#) (const char *lutfile, [QTrkSettings](#) *cfg, bool useGC, const char *name, [LocMode_t](#) locMode, int N=2000, float noiseFactor=28, float zpos=10, [ImageData](#) *pRescaledLUT=0, [vector3f](#) samplePosRange=[vector3f](#)(1, 1, 1))
- static void [ResizeLUT](#) ([ImageData](#) &lut, [ImageData](#) &resized, [QTrkSettings](#) *cfg)
- template<typename T >
std::vector< T > [logspace](#) (T a, T b, int N)
- template<typename T >
std::vector< T > [linspace](#) (T a, T b, int N)
- static [SpeedAccResult](#) [SpeedAccTest](#) ([ImageData](#) &lut, [QTrkSettings](#) *cfg, int N, [vector3f](#) centerpos, [vector3f](#) range, const char *name, int MaxPixelValue, int extraFlags=0, int buildLUTFlags=0)

11.3.1 Function Documentation

11.3.1.1 static void [EnableGainCorrection](#) ([QueuedTracker](#) * qtrk) [static]

```
190 {
191     int nb, _pl, _r;
192     qtrk->GetRadialZLUTSize(nb, _pl, _r);
193
194     int w=qtrk->cfg.width, h=qtrk->cfg.height;
195     float *offset = new float [w*h*nb];
196     float *gain = new float [w*h*nb];
197     RandomFill(offset, w*h*nb, 0, 0.1f);
198     RandomFill(gain, w*h*nb, 1, 0.2f);
199     qtrk->SetPixelCalibrationImages(offset, gain);
200     delete[] offset; delete[] gain;
201 }
```

11.3.1.2 static std::vector<vector3f> FetchResults(QueuedTracker * trk) [static]

```

69 {
70     int N = trk->GetResultCount();
71     std::vector<vector3f> results (N);
72     for (int i=0;i<N;i++) {
73         LocalizationResult j;
74         trk->FetchResults(&j,1);
75         results[j.job.frame] = j.pos;
76     }
77     return results;
78 }
```

11.3.1.3 template<typename TrackerType > std::vector<vector3f> Gauss2DTest(int NumImages = 10, int JobsPerImg = 1000, bool useGC = false)

```

211 {
212     QTrkSettings cfg;
213     cfg.width = cfg.height = 20;
214     cfg.gauss2D_iterations = 8;
215     LocMode_t lt = LT_Gaussian2D;
216     TrackerType qtrk(cfg);
217     std::vector<float*> images;
218
219     srand(0);
220
221     qtrk.SetRadialZLUT(0, 1, 1); // need to indicate 1 bead, as the pixel calibration images are
222     // per-bead
223     if (useGC) EnableGainCorrection(&qtrk);
224
225     // Schedule images to localize on
226     dbgprintf("Gauss2D: Generating %d images...\n", NumImages);
227     std::vector<float> truepos(NumImages*2);
228 //    srand(time(0));
229
230     for (int n=0;n<NumImages;n++) {
231         double t1 = GetPreciseTime();
232         float xp = cfg.width/2+(rand_uniform<float>() - 0.5) * 5;
233         float yp = cfg.height/2+(rand_uniform<float>() - 0.5) * 5;
234         truepos[n*2+0] = xp;
235         truepos[n*2+1] = yp;
236
237         float *image = new float[cfg.width*cfg.height];
238         images.push_back(image);
239
240         ImageData img(image,cfg.width,cfg.height);
241         GenerateGaussianSpotImage(&img, vector2f(xp,yp), cfg.
242         gauss2D_sigma, 1000, 4);
243         ApplyPoissonNoise(img, 1.0f);
244
245         FloatToJPEGFile(SPRINTF("gauss2d-%d.jpg", n).c_str(), image,cfg.
246         width,cfg.height);
247     }
248
249     // Measure speed
250     dbgprintf("Localizing on %d images...\n", NumImages*JobsPerImg);
251     double tstart = GetPreciseTime();
252
253     qtrk.SetLocalizationMode(lt);
254     for (int n=0;n<NumImages;n++) {
255         for (int k=0;k<JobsPerImg;k++) {
256             LocalizationJob job(n,0,0,0);
257             qtrk.ScheduleLocalization((uchar*)images[n], cfg.width*sizeof(float),
258             QTrkFloat, &job);
259         }
260     }
261
262     qtrk.Flush();
263
264     int total = NumImages*JobsPerImg;
265     int rc = qtrk.GetResultCount(), displayrc=0;
266     do {
267         rc = qtrk.GetResultCount();
268         while (displayrc<rc) {
269             if( displayrc%JobsPerImg==0) dbgprintf("Done: %d / %d\n", displayrc, total);
270             displayrc++;
271         }
272         Sleep(10);
273     } while (rc != total);
```

```

271     double tend = GetPreciseTime();
272
273     // Wait for last jobs
274     rc = NumImages*JobsPerImg;
275     double errX=0.0, errY=0.0;
276
277     std::vector<vector3f> results (NumImages);
278     while(rc>0) {
279         LocalizationResult result;
280
281         if (qtrk.FetchResults(&result, 1)) {
282             int iid = result.job.frame;
283             float x = fabs(truepos[iid*2+0]-result.pos.x);
284             float y = fabs(truepos[iid*2+1]-result.pos.y);
285
286             results [result.job.frame] = result.pos;
287
288             errX += x; errY += y;
289             rc--;
290         }
291     }
292     dbprintf("Localization Speed: %d (img/s), using %d threads\n", (int)( total/(tend-tstart) ), qtrk.cfg.numThreads);
293     dbprintf("ErrX: %f, ErrY: %f\n", errX/total, errY/total);
294     DeleteAllElems(images);
295     return results;
296 }
297 }
```

11.3.1.4 template<typename T > std::vector<T> linspace (T a, T b, int N)

```

462 {
463     std::vector<T> r (N);
464     for (int i=0;i<N;i++)
465         r[i] = (b - a) * (i/(float)(N-1)) + a;
466     return r;
467 }
```

11.3.1.5 template<typename T > std::vector<T> logspace (T a, T b, int N)

```

449 {
450     std::vector<T> r (N);
451
452     T a_ = log(a);
453     T b_ = log(b);
454
455     for (int i=0;i<N;i++)
456         r[i] = exp( (b_ - a_) * (i/(float)(N-1)) + a_);
457     return r;
458 }
```

11.3.1.6 static void MeanStDevError (const std::vector< vector3f > & truepos, const std::vector< vector3f > & v, vector3f & meanErr, vector3f & stdev) [static]

```

319 {
320     meanErr=vector3f();
321     for (size_t i=0;i<v.size();i++) meanErr+=v[i]-truepos[i];
322     meanErr*=1.0f/v.size();
323
324     vector3f r;
325     for (uint i=0;i<v.size();i++) {
326         vector3f d = (v[i]-truepos[i])-meanErr;
327         r+= d*d;
328     }
329     stdev = sqrt(r/v.size());
330 }
```

11.3.1.7 static void RandomFill (float * *d*, int *size*, float *mean*, float *std_deviation*) [static]

```

82 {
83     for (int i=0;i<size;i++)
84         d [i] = std::max(0.0f, mean + rand_normal<float>() * std_deviati
85 }

```

11.3.1.8 template<typename T > void ResampleLUT (T * *qtrk*, ImageData * *lut*, int *zplanes*, ImageData * *newlut*, const char * *jpgfile* = 0, uint *buildLUTFlags* = 0)

```

36 {
37     QTrkComputedConfig& cfg = qtrk->cfg;
38     ImageData img = ImageData::alloc(cfg.width,cfg.
39                                     height);
40     qtrk->SetRadialZLUT(0, 1, zplanes);
41     qtrk->BeginLUT(buildLUTflags);
42     for (int i=0;i<zplanes;i++)
43     {
44         vector2f pos(cfg.width/2,cfg.height/2);
45         GenerateImageFromLUT(&img, lut, qtrk->cfg.zlut_minradius,
46                               qtrk->cfg.zlut_maxradius, vector3f(pos.x,pos.y, i/(float)zplanes * lut->
47 h), true);
48         img.normalize();
49         if (i == zplanes/2 && jpgfile)
50             WriteJPEGFile(Sprintf("smp-%s",jpgfile).c_str(), img);
51         qtrk->BuildLUT(img.data, sizeof(float)*img.w, QTrkFloat, i, &pos);
52     }
53     qtrk->FinalizeLUT();
54
55     *newlut = ImageData::alloc(cfg.zlut_radialsteps, zplanes);
56     qtrk->GetRadialZLUT(newlut->data);
57     newlut->normalize();
58
59     img.free();
60
61     if (jpgfile) {
62         float* zlut_result=new float[zplanes*cfg.zlut_radialsteps*1];
63         qtrk->GetRadialZLUT(zlut_result);
64         FloatToJPEGFile(jpgfile, zlut_result, cfg.
65                         zlut_radialsteps, zplanes);
66         delete[] zlut_result;
67     }
68 }

```

11.3.1.9 static void ResizeLUT (ImageData & *lut*, ImageData & *resized*, QTrkSettings * *cfg*) [static]

```

421 {
422     QTrkComputedConfig cc(*cfg);
423
424     int nsmp = (int) ceil( (float)lut.w/cc.zlut_radialsteps );
425     resized = ImageData::alloc(cc.zlut_radialsteps, lut.h);
426     float *c = ALLOCA_ARRAY(float, cc.zlut_radialsteps);
427     for (int y=0;y<lut.h;y++) {
428         for (int i=0;i<cc.zlut_radialsteps;i++) c[i]=0;
429         for(int x=0;x<lut.w;x++) {
430             float pos = x/(float)lut.w * cc.zlut_radialsteps;
431             float frac= (int)pos - pos;
432             int p=(int)pos;
433             float v=lut.at(x,y);
434             c[p] += frac;
435             resized.at(p,y) += frac*v;
436             if (p<cc.zlut_radialsteps-1) {
437                 c[p+1] += 1-frac;
438                 resized.at(p+1,y) += (1-frac)*v;
439             }
440         }
441         for (int x=0;x<cc.zlut_radialsteps;x++)
442             resized.at(x,y) /= c[x];
443     }
444 }

```

11.3.1.10 template<typename TrkType > RunTrackerResults RunTracker (const char * *lutfile*, QTrkSettings * *cfg*, bool *useGC*, const char * *name*, LocMode_t *locMode*, int *N* = 2000, float *noiseFactor* = 28, float *zpos* = 10, ImageData * *pRescaledLUT* = 0, vector3f *samplePosRange* = vector3f(1, 1, 1))

```

351 {
352     std::vector<vector3f> results, truepos;
353
354     ImageData lut = ReadJPEGFile(lutfile);
355     ImageData img = ImageData::alloc(cfg->width, cfg->
356                                     height);
356
357     ImageData* rescaledLUT;
358     ImageData rescaledBuffer;
359
360     TrkType trk(*cfg);
361     if (pRescaledLUT) {
362         rescaledLUT = pRescaledLUT;
363         if (rescaledLUT->data == 0) {
364             *rescaledLUT = ImageData::alloc(cfg->width, cfg->
365                                             height);
365             ResampleLUT(&trk, &lut, lut.h, rescaledLUT, SPrintf("%s-zlut.jpg", name).c_str(),
366                         ((locMode&LT_FourierLUT) ? BUILDLUT_FOURIER : 0));
366         }
367     } else {
368         rescaledBuffer = ImageData::alloc(cfg->width, cfg->
369                                         height);
369         rescaledLUT = &rescaledBuffer;
370         ResampleLUT(&trk, &lut, lut.h, rescaledLUT, SPrintf("%s-zlut.jpg", name).c_str(),
371                     ((locMode&LT_FourierLUT) ? BUILDLUT_FOURIER : 0));
371     }
372
373     if (useGC) EnableGainCorrection(&trk);
374
375 //trk.SetConfigValue("use_texturecache", "0");
376
377 srand(0);
378 trk.SetLocalizationMode(locMode);
379 for (int i=0;i<N;i++)
380 {
381     vector3f pos(
382         cfg->width/2 + samplePosRange.x*(rand_uniform<float>() - 0.5f),
383         cfg->height/2 + samplePosRange.y*(rand_uniform<float>() - 0.5f),
384         zpos + samplePosRange.z*(rand_uniform<float>() - 0.5f)
385     );
386     GenerateImageFromLUT(&img, rescaledLUT, trk.cfg.zlut_minradius, trk.cfg.
387                           zlut_maxradius, vector3f( pos.x, pos.y, pos.z));
388     if (noiseFactor>0) ApplyPoissonNoise(img, noiseFactor * 255, 255);
389     truepos.push_back(pos);
390
391     LocalizationJob job(i, 0, 0, 0);
392     trk.ScheduleImageData(&img, &job);
393 }
394
395 dbgprintf("Scheduled %d images\n", N);
396
397 WaitForFinish(&trk, N);
398
399 results.resize(trk.GetResultCount());
400 for (uint i=0;i<results.size();i++) {
401     LocalizationResult r;
402     trk.FetchResults(&r,1);
403     results[r.job.frame]=r.pos;
404 }
405
406 double tend = GetPreciseTime();
407 img.free();
408 lut.free();
409
410 RunTrackerResults r;
411 r.output=results;
412 r.truepos=truepos;
413
414 if (!pRescaledLUT)
415     rescaledBuffer.free();
416
417 return r;
418 }
```

11.3.1.11 static SpeedAccResult SpeedAccTest (ImageData & *lut*, QTrkSettings * *cfg*, int *N*, vector3f *centerpos*, vector3f *range*, const char * *name*, int *MaxPixelValue*, int *extraFlags* = 0, int *buildLUTFlags* = 0) [static]

```
497 {
```

```

498     typedef QueuedTracker TrkType;
499     int NImg=N; //std::max(1,N/20);
500     std::vector<vector3f> results, truepos (NImg);
501
502     std::vector<ImageData> imgs (NImg);
503 //   const float R=5;
504
505     QueuedTracker* trk = new QueuedCPUTracker (*cfg); // CreateQueuedTracker(*cfg);
506
507     ImageData resizedLUT;
508     ResampleLUT(trk, &lut, lut.h, &resizedLUT, 0, buildLUTFlags);
509
510     if (buildLUTFlags&BUILD_LUT_BIASCORRECT)
511         trk->ComputeZBiasCorrection(lut.h*10, 0, 4, true);
512
513     std::vector<Matrix3X3> fishers (NImg);
514
515 //   for (int i=0;i<NImg;i++) {
516     parallel_for(NImg, [&](int i) {
517         imgs[i]=ImageData::alloc(cfg->width,cfg->height);
518         vector3f pos = centerpos + range*vector3f(rand_uniform<float>()-0.5f,
519             rand_uniform<float>()-0.5f, rand_uniform<float>()-0.5f)*1;
520         GenerateImageFromLUT(&imgs[i], &resizedLUT, trk->cfg.
521             zlut_minradius, trk->cfg.zlut_maxradius, vector3f( pos.
522             x, pos.y, pos.z));
523
524         SampleFisherMatrix fm(MaxPixelValue);
525         fishers[i] = fm.Compute(pos, vector3f(1,1,1)*0.001f, resizedLUT, cfg->
526             width,cfg->height, trk->cfg.zlut_minradius,trk->cfg.
527             zlut_maxradius);
528
529         imgs[i].normalize();
530         if (MaxPixelValue> 0) ApplyPoissonNoise(imgs[i], MaxPixelValue);
531         if(i==0) WriteJPEGFile(name, imgs[i]);
532
533         truepos[i]=pos;
534     });
535
536     Matrix3X3 fisher;
537     for (int i=0;i<NImg;i++) fisher+=fishers[i];
538
539     int flags= LT_LocalizeZ|LT_NormalizeProfile|extraFlags;
540     if (cfg->qi_iterations>0) flags|=LT_QI;
541
542     trk->SetLocalizationMode((LocMode_t)flags);
543     double tstart=GetPreciseTime();
544
545     for (int i=0;i<N;i++)
546     {
547         LocalizationJob job(i, 0, 0, 0);
548         trk->ScheduleLocalization((uchar*)imgs[i%NImg].data, sizeof(float)*cfg->
549             width, QTrkFloat, &job);
550     }
551
552     WaitForFinish(trk, N);
553     double tend = GetPreciseTime();
554
555     results.resize(trk->GetResultCount());
556     for (uint i=0;i<results.size();i++) {
557         LocalizationResult r;
558         trk->FetchResults(&r,1);
559         results[r.job.frame]=r.pos;
560     }
561
562     for (int i=0;i<NImg;i++)
563         imgs[i].free();
564
565     SpeedAccResult r;
566     r.Compute(results, [&](int index) { return truepos[index]; });
567
568     r.speed = N/(tend-tstart);
569     fisher *= 1.0f/NImg;
570     r.crlb = sqrt(fisher.Inverse().diag());
571     resizedLUT.free();
572     delete trk;
573     return r;
574 }

```

11.3.1.12 template<typename TrackerType > void TestCMOSNoiseInfluence (const char * lutfile)

```

97     int nNoiseLevels = 5;
98     float gainStDevScale = 0.3f;
99 #ifdef _DEBUG
100    int nDriftSteps = 100;
101 #else
102    int nDriftSteps = 2000;
103 #endif
104    float driftDistance = 4; // pixels
105
106    ImageData lut = ReadJPEGFile(lutfile);
107    ImageData img = ImageData::alloc(120,120);
108
109    QTrkSettings cfg;
110    cfg.width = img.w; cfg.height = img.h;
111    TrackerType trk(cfg);
112
113    int nBeads=1;
114    float zlutMinR=4, zlutMaxR = cfg.width/2-5;
115    ResampleLUT(&trk, &lut, zlutMinR, zlutMaxR, 100, "resample-zlut.jpg");
116
117    float *offset = new float [nBeads * cfg.width * cfg.height];
118    float *gain = new float [nBeads * cfg.width * cfg.height];
119
120    bool useCalib = false;
121    bool saveImgs = true;
122
123    float *info = new float [nNoiseLevels*2];
124
125    trk.SetLocalizationMode((LocMode_t) (LT_QI|LT_LocalizeZ|
LT_NormalizeProfile));
126
127    for (int nl=0;nl<nNoiseLevels;nl++) {
128
129        float offset_stdev = info[nl*2+0] = nl*0.01f;
130        float gain_stdev = info[nl*2+1] = nl*gainStDevScale;
131
132        srand(0);
133        RandomFill(offset, nBeads * cfg.width * cfg.height, 0, offset_stdev);
134        RandomFill(gain, nBeads * cfg.width * cfg.height, 1, gain_stdev);
135
136        if (useCalib) trk.SetPixelCalibrationImages(offset, gain);
137
138        std::string dirname= SPrintf("noiselev%d", nl);
139        if (saveImgs) _mkdir(dirname.c_str());
140
141        // drift
142        float dx = driftDistance / (nDriftSteps-1);
143        for (int d=0;d<nDriftSteps;d++) {
144            GenerateImageFromLUT(&img, &lut, zlutMinR, zlutMaxR,
vector3f(img.w/2+dx*d,img.h/2, 20));
145            img.normalize();
146
147            if (!useCalib) {
148                for (int i=0;i<cfg.width*cfg.height;i++)
149                    img[i]*=gain[i];
150                ApplyPoissonNoise(img, 255);
151                for (int i=0;i<cfg.width*cfg.height;i++)
152                    img[i]+=offset[i];
153            } else
154                ApplyPoissonNoise(img, 255);
155
156            if (saveImgs && d<40) {
157                FloatToJPEGFile(SPrintf("%s\\nl%d-smp%d.jpg",dirname.c_str(),nl,d).
c_str(),img.data, img.w,img.h);
158            }
159
160            if (d==0) {
161                FloatToJPEGFile(SPrintf("nl%d.jpg",nl).c_str(),img.
data, img.w,img.h);
162            }
163
164            if ( d%(std::max(1,nDriftSteps/10)) == 0 )
165                dbgprintf("Generating images for test %d...\\n", d);
166
167            LocalizationJob job(d, d, 0, 0);
168            trk.ScheduleLocalization((uchar*)img.data, sizeof(float)*img.
w, QTrkFloat, &job);
169        }
170        trk.Flush();
171
172        while (!trk.IsIdle());
173        int nResults = trk.GetResultCount();
174        dbgprintf("noiselevel: %d done. Writing %d results\\n", nl, nResults);
175        auto results = FetchResults(&trk);
176        WriteTrace(SPrintf("%s\\trace.txt", dirname.c_str()), &results[0], results.size())
177    ;

```

```

178     }
179     WriteImageAsCSV("offset_gain_stdev.txt", info, 2, nNoiseLevels);
180
181     img.free();
182     lut.free();
183
184     delete[] gain;
185     delete[] offset;
186     delete[] info;
187 }
```

11.3.1.13 static double WaitForFinish (QueuedTracker * qtrk, int N) [static]

```

301 {
302     double t0 = GetPreciseTime();
303     qtrk->Flush();
304     int displayrc=0,rc=0;
305     while ( (rc = qtrk->GetResultCount ()) != N || displayrc<rc) {
306         while (displayrc<rc) {
307             if(displayrc%std::max(1,N/10)==0) dbgprintf("Done: %d / %d\n", displayrc, N);
308             displayrc++;
309         }
310         Threads::Sleep(10);
311     }
312     dbgprintf("Done: %d / %d\n", displayrc, N);
313     double t1 = GetPreciseTime();
314     return t1-t0;
315 }
```

11.4 cputrack-test/testutils.cpp File Reference

```
#include <time.h>
#include <string>
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <sstream>
#include "testutils.h"
```

Functions

- float **distance** (**vector2f** a, **vector2f** b)
- bool **DirExists** (const std::string &dirName_in)
- int **NumFilesInDir** (const std::string &dirName_in)
- int **NumJpgInDir** (const std::string &dirName_in)
- **ImageData** **CropImage** (**ImageData** img, int x, int y, int w, int h)
- **ImageData** **ResizeImage** (**ImageData** img, int factor)
- **ImageData** **AddImages** (**ImageData** img1, **ImageData** img2, **vector2f** displacement)
- **ImageData** **GaussMask** (**ImageData** img, float sigma)
- **ImageData** **SkewImage** (**ImageData** img, float fact)
- void **GetOuterEdges** (float *out, int size, **ImageData** img)
- float **BackgroundMedian** (**ImageData** img)
- float **BackgroundStdDev** (**ImageData** img)
- float **BackgroundRMS** (**ImageData** img)

11.4.1 Function Documentation

11.4.1.1 `ImageData AddImages (ImageData img1, ImageData img2, vector2f displacement)`

```

165 {
166     ImageData addedImg = ImageData::alloc(img1.w,img1.
167     h);
168     for(int x_i=0;x_i

```

11.4.1.2 `float BackgroundMedian (ImageData img)`

```

235     {
236     int size = img.w * 2 + img.h * 2 - 4;
237     float* outeredge = new float[size];
238     GetOuterEdges(outeredge,size,img);
239     std::sort(outeredge,outeredge+size);
240     float median;
241     if(size % 2 == 0)
242         median = (outeredge[(int)(size/2-1)] + outeredge[(int)(size/2+1)])/2;
243     else
244         median = outeredge[size/2];
245     delete[] outeredge;
246     return median;
247 }
```

11.4.1.3 `float BackgroundRMS (ImageData img)`

```

258     {
259     int size = img.w * 2 + img.h * 2 - 4;
260     float* outeredge = new float[size];
261     GetOuterEdges(outeredge,size,img);
262     float sqsum = 0.0f;
263     for(int ii = 0; ii < size; ii++){
264         sqsum += outeredge[ii]*outeredge[ii];
265     }
266     delete[] outeredge;
267     return sqrt(1/(float)size*sqsum);
268 }
```

11.4.1.4 `float BackgroundStdDev (ImageData img)`

```

249     {
250     int size = img.w * 2 + img.h * 2 - 4;
251     float* outeredge = new float[size];
252     GetOuterEdges(outeredge,size,img);
253     float stddev = ComputeStdDev(outeredge,size);
254     delete[] outeredge;
255     return stddev;
256 }
```

11.4.1.5 **ImageData CropImage (ImageData img, int x, int y, int w, int h)**

```

133 {
134     ImageData croppedImg = ImageData::alloc(w,h);
135
136     if( x < 0 || y < 0 || x + w > img.w || y + h > img.h){
137         return img;
138     }
139
140     for(int x_i = x; x_i < x+w; x_i++){
141         for(int y_i = y; y_i < y+h; y_i++){
142             croppedImg.at(x_i-x,y_i-y) = img.at(x_i,y_i);
143         }
144     }
145     return croppedImg;
146 }
```

11.4.1.6 **bool DirExists (const std::string & dirName_in)**

```

13 {
14     DWORD ftyp = GetFileAttributesA(dirName_in.c_str());
15     if (ftyp == INVALID_FILE_ATTRIBUTES)
16         return false; //something is wrong with your path!
17
18     if (ftyp & FILE_ATTRIBUTE_DIRECTORY)
19         return true; // this is a directory!
20
21     return false; // this is not a directory!
22 }
```

11.4.1.7 **float distance (vector2f a, vector2f b)**

```
10 { return distance(a.x-b.x,a.y-b.y); }
```

11.4.1.8 **ImageData GaussMask (ImageData img, float sigma)**

```

181 {
182     ImageData gaussImg = ImageData::alloc(img.w,img.h);
183     vector2f centre = vector2f(img.w/2,img.h/2);
184     for(int x_i=0;x_i

```

11.4.1.9 **void GetOuterEdges (float * out, int size, ImageData img)**

```

212
213     int x,y=0;
214     for(int ii = 0; ii < size; ii++){
215         if(ii < img.w){ // Top
216             x = ii;
217             y = 0;
218         }
219         else if(ii < img.w + img.h - 1){ // Right
220             x = img.w-1;
221             y = ii-(img.w-1);
222         }
223         else if(ii < img.w * 2 + img.h - 2){ // Bottom
224             y = img.h-1;
225             x = ii-(img.h+img.w-1);
226         }
227         else{ // Left
228             x = 0;
229             y = ii-(img.h+img.w*2-3);
230         }
231         out[ii] = img.at(x,y);
232     }
233 }
```

11.4.1.10 int NumFilesInDir (const std::string & dirName_in)

```

25 {
26     WIN32_FIND_DATA FindFileData;
27     HANDLE hFind;
28
29     std::string dirName = dirName_in + "*";
30     hFind = FindFirstFile(dirName.c_str(),&FindFileData);
31     if (hFind == INVALID_HANDLE_VALUE) {
32         printf ("FindFirstFile failed (%d)\n", GetLastError());
33         return 0;
34     }
35     int out = -1;
36     while(FindNextFile(hFind,&FindFileData)) {
37         out++;
38     }
39     return out;
40 }
```

11.4.1.11 int NumJpgInDir (const std::string & dirName_in)

```

43 {
44     WIN32_FIND_DATA FindFileData;
45     HANDLE hFind;
46
47     std::string dirName = dirName_in + "*";
48     hFind = FindFirstFile(dirName.c_str(),&FindFileData);
49     if (hFind == INVALID_HANDLE_VALUE) {
50         printf ("FindFirstFile failed (%d)\n", GetLastError());
51         return 0;
52     }
53     int out = -1;
54     while(FindNextFile(hFind,&FindFileData)) {
55         std::string name = FindFileData.cFileName;
56         if(name.find(".jpg")!=namenpos) {
57             out++;
58         }
59     }
60     return out;
61 }
```

11.4.1.12 ImageData ResizeImage (ImageData img, int factor)

```

149 {
150     ImageData resizedImg = ImageData::alloc(img.w*factor,img.
151 h*factor);
152     for(int x_i=0;x_i

```

11.4.1.13 ImageData SkewImage (ImageData img, float fact)

```

195 {
196     ImageData skewImg = ImageData::alloc(img.w,img.h);
197     vector2f centre = vector2f(img.w/2,img.h/2);
198     float median = BackgroundMedian(img);
199     float stddev = BackgroundStdDev(img);
200     float maxskew = median*stddev;
201     for(int x_i=0;x_i

```

11.5 cputrack-test/testutils.h File Reference

```
#include "std_incl.h"
#include "QueuedCPUTracker.h"
```

Classes

- class [outputter](#)
- struct [outputter::outputModes](#)

Enumerations

- enum [OutputModes](#) { [Console](#) = 1, [Files](#) = 2, [Images](#) = 4 }

Functions

- template<typename T >
 T [sq](#) (T x)
- template<typename T >
 T [distance](#) (T x, T y)
- bool [DirExists](#) (const std::string &dirName_in)
- int [NumFilesInDir](#) (const std::string &dirName_in)
- int [NumJpgInDir](#) (const std::string &dirName_in)
- float [distance](#) (vector2f a, vector2f b)
- [ImageData](#) [CropImage](#) ([ImageData](#) img, int x, int y, int w, int h)
- [ImageData](#) [ResizeImage](#) ([ImageData](#) img, int factor)
- [ImageData](#) [AddImages](#) ([ImageData](#) img1, [ImageData](#) img2, vector2f displacement)
- [ImageData](#) [GaussMask](#) ([ImageData](#) img, float sigma)
- [ImageData](#) [SkewImage](#) ([ImageData](#) img, float fact)
- void [GetOuterEdges](#) (float *out, int size, [ImageData](#) img)
- float [BackgroundMedian](#) ([ImageData](#) img)
- float [BackgroundStdDev](#) ([ImageData](#) img)
- float [BackgroundRMS](#) ([ImageData](#) img)

11.5.1 Enumeration Type Documentation

11.5.1.1 enum OutputModes

Enumerator

- Console***
Files
Images

```
15      {
16      Console = 1,
17      Files = 2,
18      Images = 4
19  };
```

11.5.2 Function Documentation

11.5.2.1 `ImageData AddImages (ImageData img1, ImageData img2, vector2f displacement)`

```

165 {
166     ImageData addedImg = ImageData::alloc(img1.w,img1.
167     h);
168     for(int x_i=0;x_i

```

11.5.2.2 `float BackgroundMedian (ImageData img)`

```

235     {
236     int size = img.w * 2 + img.h * 2 - 4;
237     float* outeredge = new float[size];
238     GetOuterEdges(outeredge,size,img);
239     std::sort(outeredge,outeredge+size);
240     float median;
241     if(size % 2 == 0)
242         median = (outeredge[(int)(size/2-1)] + outeredge[(int)(size/2+1)])/2;
243     else
244         median = outeredge[size/2];
245     delete[] outeredge;
246     return median;
247 }
```

11.5.2.3 `float BackgroundRMS (ImageData img)`

```

258     {
259     int size = img.w * 2 + img.h * 2 - 4;
260     float* outeredge = new float[size];
261     GetOuterEdges(outeredge,size,img);
262     float sqsum = 0.0f;
263     for(int ii = 0; ii < size; ii++){
264         sqsum += outeredge[ii]*outeredge[ii];
265     }
266     delete[] outeredge;
267     return sqrt(1/(float)size*sqsum);
268 }
```

11.5.2.4 `float BackgroundStdDev (ImageData img)`

```

249     {
250     int size = img.w * 2 + img.h * 2 - 4;
251     float* outeredge = new float[size];
252     GetOuterEdges(outeredge,size,img);
253     float stddev = ComputeStdDev(outeredge,size);
254     delete[] outeredge;
255     return stddev;
256 }
```

11.5.2.5 `ImageData CropImage (ImageData img, int x, int y, int w, int h)`

```

133 {
134     ImageData croppedImg = ImageData::alloc(w,h);
135
136     if( x < 0 || y < 0 || x + w > img.w || y + h > img.h){
137         return img;
138     }
139
140     for(int x_i = x; x_i < x+w; x_i++){
141         for(int y_i = y; y_i < y+h; y_i++){
142             croppedImg.at(x_i-x,y_i-y) = img.at(x_i,y_i);
143         }
144     }
145     return croppedImg;
146 }
```

11.5.2.6 `bool DirExists (const std::string & dirName_in)`

```

13 {
14     DWORD ftyp = GetFileAttributesA(dirName_in.c_str());
15     if (ftyp == INVALID_FILE_ATTRIBUTES)
16         return false; //something is wrong with your path!
17
18     if (ftyp & FILE_ATTRIBUTE_DIRECTORY)
19         return true; // this is a directory!
20
21     return false; // this is not a directory!
22 }
```

11.5.2.7 `template<typename T> T distance (T x, T y)`

```
7 { return sqrt(x*x+y*y); }
```

11.5.2.8 `float distance (vector2f a, vector2f b)`

```
10 { return distance(a.x-b.x,a.y-b.y); }
```

11.5.2.9 `ImageData GaussMask (ImageData img, float sigma)`

```

181 {
182     ImageData gaussImg = ImageData::alloc(img.w,img.h);
183     vector2f centre = vector2f(img.w/2,img.h/2);
184     for(int x_i=0;x_i

```

11.5.2.10 void GetOuterEdges (float * out, int size, ImageData img)

```

212
213     int x,y=0;
214     for(int ii = 0; ii < size; ii++){
215         if(ii < img.w){ // Top
216             x = ii;
217             y = 0;
218         }
219         else if(ii < img.w + img.h - 1){ // Right
220             x = img.w-1;
221             y = ii-(img.w-1);
222         }
223         else if(ii < img.w * 2 + img.h - 2){ // Bottom
224             y = img.h-1;
225             x = ii-(img.h+img.w-1);
226         }
227         else{ // Left
228             x = 0;
229             y = ii-(img.h+img.w*2-3);
230         }
231         out[ii] = img.at(x,y);
232     }
233 }
```

11.5.2.11 int NumFilesInDir (const std::string & dirName_in)

```

25 {
26     WIN32_FIND_DATA FindFileData;
27     HANDLE hFind;
28
29     std::string dirName = dirName_in + "*";
30     hFind = FindFirstFile(dirName.c_str(),&FindFileData);
31     if (hFind == INVALID_HANDLE_VALUE) {
32         printf ("FindFirstFile failed (%d)\n", GetLastError());
33         return 0;
34     }
35     int out = -1;
36     while(FindNextFile(hFind,&FindFileData)){
37         out++;
38     }
39     return out;
40 }
```

11.5.2.12 int NumJpgInDir (const std::string & dirName_in)

```

43 {
44     WIN32_FIND_DATA FindFileData;
45     HANDLE hFind;
46
47     std::string dirName = dirName_in + "*";
48     hFind = FindFirstFile(dirName.c_str(),&FindFileData);
49     if (hFind == INVALID_HANDLE_VALUE) {
50         printf ("FindFirstFile failed (%d)\n", GetLastError());
51         return 0;
52     }
53     int out = -1;
54     while(FindNextFile(hFind,&FindFileData)){
55         std::string name = FindFileData.cFileName;
56         if(name.find(".jpg") !=namenpos) {
57             out++;
58         }
59     }
60     return out;
61 }
```

11.5.2.13 **ImageData ResizeImage (ImageData img, int factor)**

```

149 {
150     ImageData resizedImg = ImageData::alloc(img.w*factor, img.
151     h*factor);
151
152     for(int x_i=0;x_i

```

11.5.2.14 **ImageData SkewImage (ImageData img, float fact)**

```

195 {
196     ImageData skewImg = ImageData::alloc(img.w, img.h);
197     vector2f centre = vector2f(img.w/2, img.h/2);
198     float median = BackgroundMedian(img);
199     float stddev = BackgroundStdDev(img);
200     float maxskew = median+stddev;
201     for(int x_i=0;x_i

```

11.5.2.15 **template<typename T > T sq (T x)**

```
6 { return x*x; }
```

11.6 cputrack/BeadFinder.cpp File Reference

```
#include "std_incl.h"
#include "utils.h"
#include "kissfft.h"
#include <list>
#include "threads.h"
#include <functional>
#include "BeadFinder.h"
```

Functions

- int **NextPowerOf2** (int x)
- float **abs** (std::complex< float > x)
- void **ComplexToJPEGFile** (const char *name, std::complex< float > *d, int w, int h, bool logMode=false)
- void **FFT2D** (std::complex< float > *d, int w, int h, bool inverse)
- **Position SearchArea** (std::complex< float > *cimg, int w, int h, int px, int py, int dist)
- void **RecenterAndFilter** (ImageData *img, std::vector< Position > &pts, Config *cfg)

11.6.1 Function Documentation

11.6.1.1 float abs (std::complex< float > x)

```
19 { return sqrtf(x.imag()*x.imag()+x.real()*x.real()); }
```

11.6.1.2 void ComplexToJPEGFile (const char * name, std::complex< float > * d, int w, int h, bool logMode = false)

```
22 {
23     float minV=abs(d[0]),maxV=minV;
24     for(int i=0;i<w*h;i++) {
25         float v = abs(d[i]);
26         minV=std::min(minV, v);
27         maxV=std::max(maxV, v);
28     }
29     if (maxV==minV) {
30         dbgprintf("ComplexToJPEGFile (%s): Min=max\n",name);
31         maxV=minV+1;
32     }
33
34     uint8_t *img = new uint8_t[w*h];
35     for (int i=0;i<w*h;i++) {
36         float v = abs(d[i]);
37         img[i] = (uint8_t) (0.5f + (logMode?10:1) * 255 * (v-minV)/(maxV-minV) );
38     }
39     WriteJPEGFile(img, w,h,name,99);
40     delete[] img;
41 }
```

11.6.1.3 void FFT2D (std::complex< float > * d, int w, int h, bool inverse)

```
45 {
46     kissfft<float> xfft(w, inverse);
47     kissfft<float> yfft(h, inverse);
48
49     auto fx = [&] (int y) {
50         std::complex<float>* tmp = ALLOCA_ARRAY(std::complex<float>, std::max(w,h));
51         xfft.transform(&d[w*y], tmp);
52         for(int x=0;x<w;x++)
53             d[w*y+x]=tmp[x];
54     };
55     ThreadPool<int, std::function< void (int index) > >
56     xfftPool(fx);
57     for (int y=0;y<h;y++)
58         xfftPool.AddWork(y);
59     xfftPool.WaitUntilDone();
60
61     auto fy = [&] (int x) {
62         std::complex<float>* src = ALLOCA_ARRAY(std::complex<float>, h);
63         std::complex<float>* tmp = ALLOCA_ARRAY(std::complex<float>, std::max(w,h));
64         for (int y=0;y<h;y++)
65             src[y] = d[y*w+x];
66         yfft.transform(src, tmp);
67         for(int y=0;y<h;y++)
68             d[w*y+x]=tmp[y];
69     };
70     ThreadPool<int, std::function<void (int x) > > yfftPool(fy)
71     ;
72     for (int x=0;x<w;x++)
73         yfftPool.AddWork(x);
74     yfftPool.WaitUntilDone();
75 }
```

11.6.1.4 int NextPowerOf2 (int x)

```
12 {
13     int y=1;
14     while (y<x)
15         y*=2;
16     return y;
17 }
```

11.6.1.5 void RecenterAndFilter (**ImageData** * *img*, std::vector< **Position** > & *pts*, **Config** * *cfg*)

```

105 {
106     int roi = cfg->roi;
107     std::list<Position> newlist;
108
109     for (int i=0;i<pts.size();i++) {
110
111         // Compute COM
112         for (int
113             auto bp = pts[i];
114
115             if (bp.x < 0 || bp.y < 0 || bp.x > img->w-1-roi || bp.y > img->h-1-roi)
116                 continue;
117
118             float sum = 0.0f;
119             for (int y=0;y<roi;y++)
120                 for (int x=0;x<roi;x++) {
121                     float v = img->at(x+bp.x,y+bp.y);
122                     sum += v;
123                 }
124
125             float sumX=0,sumY=0;
126             float mean = sum/(roi*roi);
127             sum=0;
128             for (int y=0;y<roi;y++) {
129                 for (int x=0;x<roi;x++) {
130                     float v = img->at(x+bp.x,y+bp.y)-mean;
131                     v=v*v;
132                     sum += v;
133                     sumX += v*x;
134                     sumY += v*y;
135                 }
136             }
137
138             if (sum==0.0f)
139                 continue;
140
141             bp.x += sumX/sum-roi/2;
142             bp.y += sumY/sum-roi/2;
143
144             // discard them if they are now getting outside of the image boundary
145             if (bp.x < 0 || bp.y < 0 || bp.x > img->w-1-roi || bp.y > img->h-1-roi)
146                 continue;
147
148             float m = cfg->MinPixelDistance();
149             bool accepted=true;
150             for (auto c=newlist.begin();c!=newlist.end();++c) {
151                 int dx=c->x-bp.x, dy=c->y-bp.y;
152                 if (dx*dx+dy*dy<m*m) {
153                     // bp is very close to *c, so they're probably both shitty beads
154                     newlist.erase(c);
155                     accepted=false;
156                     break;
157                 }
158             }
159
160             if (accepted)
161                 newlist.push_back(bp);
162         }
163
164         pts.clear();
165         pts.insert(pts.begin(), newlist.begin(),newlist.end());
166     }

```

11.6.1.6 **Position** SearchArea (std::complex< float > * *cimg*, int *w*, int *h*, int *px*, int *py*, int *dist*)

```

77 {
78     Position best(px,py);
79     float bestValue = cimg[py*w+px].real();
80
81     int minY = std::max(0,py-dist);
82     int minX = std::max(0,px-dist);
83     int maxX = std::min(w-1,px+dist);
84     int maxY = std::min(h-1,py+dist);
85
86     int dist2 =dist*dist;
87     for (int y=minY;y<=maxY;y++) {
88         for (int x=minX;x<=maxX;x++) {
89             int dx=x-px,dy=y-py;
90             if (dx*dx+dy*dy<=dist2)

```

```

91             {
92                 float v = cimg[y*w+x].real();
93                 if(v > bestValue) {
94                     best = Position(x,y);
95                     bestValue = v;
96                 }
97                 cimg[y*w+x] = std::complex<float>();
98             }
99         }
100     }
101 }
```

11.7 cputrack/BeadFinder.h File Reference

```
#include "utils.h"
```

Classes

- struct [BeadFinder::Position](#)
- struct [BeadFinder::Config](#)

Namespaces

- [BeadFinder](#)

Functions

- std::vector< [Position](#) > [BeadFinder::Find](#) ([ImageData](#) *img, float *sample, [Config](#) *cfg)
- std::vector< [Position](#) > [BeadFinder::Find](#) (uint8_t *img, int pitch, int w, int h, int smpCornerX, int smpCornerY, [Config](#) *cfg)

11.8 cputrack/BenchmarkLUT.cpp File Reference

```
#include "std_incl.h"
#include "BenchmarkLUT.h"
#include "LsqQuadraticFit.h"
```

11.9 cputrack/BenchmarkLUT.h File Reference

```
#include "utils.h"
```

Classes

- class [BenchmarkLUT](#)

11.10 cputrack/cpu_tracker.cpp File Reference

```
#include "std_incl.h"
#include <exception>
#include <cmath>
#include "QueuedTracker.h"
#include "cpu_tracker.h"
#include "LsqQuadraticFit.h"
#include "random_distr.h"
#include "CubicBSpline.h"
#include "../cudatrack/simplefft.h"
#include "DebugResultCompare.h"
```

Macros

- #define ZLUT_CMPDATA
- #define SFFT_BOTH
- #define MARKPIXEL(x, y)
- #define MARKPIXELI(x, y)

Functions

- static int round (scalar_t f)
- template<typename T >
T conjugate (const T &v)
- static int clamp (int v, int a, int b)
- template<typename T >
double sum_diff (T *begin, T *end, T *other)

Variables

- const float XCorScale = 1.0f
- const scalar_t QIWeights [QI_LSQFIT_NWEIGHTS] = QI_LSQFIT_WEIGHTS
- const float ZLUTWeights [ZLUT_LSQFIT_NWEIGHTS] = ZLUT_LSQFIT_WEIGHTS
- const double ZLUTWeights_d [ZLUT_LSQFIT_NWEIGHTS] = ZLUT_LSQFIT_WEIGHTS

11.10.1 Macro Definition Documentation

11.10.1.1 #define MARKPIXEL(x, y)

11.10.1.2 #define MARKPIXELI(x, y)

11.10.1.3 #define SFFT_BOTH

11.10.1.4 #define ZLUT_CMPDATA

11.10.2 Function Documentation

11.10.2.1 static int clamp (int v, int a, int b) [static]

```
39 { return std::max(a, std::min(b, v)); }
```

11.10.2.2 template<typename T> T conjugate (const T & v)

```
33 { return T(v.real(), -v.imag()); }
```

11.10.2.3 static int round (scalar_t f) [static]

```
31 { return (int)(f+0.5f); }
```

11.10.2.4 template<typename T> double sum_diff (T * begin, T * end, T * other)

```
369 {
370     double sd = 0.0;
371     for (T* i = begin; i != end; i++, other++) {
372         T d = *i - *other;
373         sd += abs(d.real()) + abs(d.imag());
374     }
375     return sd;
376 }
```

11.10.3 Variable Documentation

11.10.3.1 const scalar_t QIWeights[QI_LSQFIT_NWEIGHTS] = QI_LSQFIT_WEIGHTS

11.10.3.2 const float XCorScale = 1.0f

11.10.3.3 const float ZLUTWeights[ZLUT_LSQFIT_NWEIGHTS] = ZLUT_LSQFIT_WEIGHTS

11.10.3.4 const double ZLUTWeights_d[ZLUT_LSQFIT_NWEIGHTS] = ZLUT_LSQFIT_WEIGHTS

11.11 cputrack/cpu_tracker.h File Reference

```
#include "QueuedTracker.h"
#include "utils.h"
#include "scalar_types.h"
#include "kissfft.h"
```

Classes

- class [XCor1DBuffer](#)
- class [CPUTracker](#)
- class [CPUTracker::FFT2D](#)
- struct [CPUTracker::Gauss2DResult](#)

TypeDefs

- typedef [uchar pixel_t](#)

Functions

- `CPUTracker * CreateCPUTrackerInstance (int w, int h, int xcorw)`

11.11.1 Typedef Documentation

11.11.1.1 `typedef uchar pixel_t`

11.11.2 Function Documentation

11.11.2.1 `CPUTracker* CreateCPUTrackerInstance (int w, int h, int xcorw)`

11.12 cputrack/CubicBSpline.h File Reference

Macros

- `#define CUDA_SUPPORTED_FUNC`

Functions

- `void CUDA_SUPPORTED_FUNC ComputeBSplineWeights (float w[], float t)`
- `template<typename T >`
`void CUDA_SUPPORTED_FUNC ComputeBSplineDerivatives (float t, T *k, T &deriv, T &deriv2)`
- `template<typename T >`
`float CUDA_SUPPORTED_FUNC ComputeSplineFitMaxPos (T *data, int len)`

11.12.1 Macro Definition Documentation

11.12.1.1 `#define CUDA_SUPPORTED_FUNC`

11.12.2 Function Documentation

11.12.2.1 `template<typename T > void CUDA_SUPPORTED_FUNC ComputeBSplineDerivatives (float t, T * k, T & deriv, T & deriv2)`

```

43 {
44     float t2=t*t;
45     float t3=t2*t;
46     float omt=1-t;
47
48     deriv = 0.5f * (-omt*omt * k[0] + (3*t2 - 4*t) * k[1] + (-3*t2 + 2*t + 1) * k[2] + t2 * k[3]);
49     deriv2 = omt * k[0] + (3*t - 2) * k[1] + (-3*t + 1) * k[2] + t * k[3];
50 }
```

11.12.2.2 void CUDA_SUPPORTED_FUNC ComputeBSplineWeights (float w[], float t) [inline]

```

16 {
17     float t2=t*t;
18     float t3=t2*t;
19     float omt = 1-t;
20     w[0] = 1.0f/6 * omt*omt*omt;
21     w[1] = 1.0f/6 * (3*t3 - 6*t2 + 4);
22     w[2] = 1.0f/6 * (-3*t3 + 3*t2 + 3*t + 1);
23     w[3] = 1.0f/6 * t3;
24 }

```

11.12.2.3 template<typename T > float CUDA_SUPPORTED_FUNC ComputeSplineFitMaxPos (T * data, int len)

```

56 {
57     // find the maximum by doing following the spline gradient
58     int iMax=0;
59     T vMax=data[0];
60     for (int k=1;k<len;k++) {
61         if (data[k]>vMax) {
62             vMax = data[k];
63             iMax = k;
64         }
65     }
66     if (iMax < 1) iMax=1;
67     if (len < 4) return iMax;
68
69     // x(t) = w(0, t) * p(0) + w(1, t) * p(1) + w(2, t) * p(2) + w(3, t) * p(3)
70     // x'(t) = w'(0, t) * p(0) + w'(1, t) * p(1) + w'(2, t) * p(2) + w'(3, t) * p(3)
71     // x''(t) = w''(0, t) * p(0) + w''(1, t) * p(1) + w''(2, t) * p(2) + w''(3, t) * p(3)
72
73     T x = iMax;
74     for(int it=0;it<5;it++) {
75         if (x < 1 )x = 1;
76         if (x >= len-2) x = len-2;
77         int i = (int)x;
78         T t=x-i;
79         T w,w2;
80         ComputeBSplineDerivatives(t, &data[i-1], w, w2);
81         T dx=w/w2;
82 //         dbgprintf("[%d]: x=%f, w=%f, w2=%f. dx=%f\n", it, x,w,w2, -dx);
83         x -= dx;
84     }
85     if (x<0) x=0;
86     if (x>=len-1) x=len-1;
87     return x;
88 }

```

11.13 cputrack/DebugResultCompare.h File Reference

11.14 cputrack/dllmacros.h File Reference

Macros

- #define DLL_CALLCONV __cdecl
- #define DLL_EXPORT __declspec(dllexport)
- #define CDLL_EXPORT extern "C" DLL_EXPORT

11.14.1 Macro Definition Documentation

11.14.1.1 `#define CDLL_EXPORT extern "C" DLL_EXPORT`

11.14.1.2 `#define DLL_CALLCONV __cdecl`

11.14.1.3 `#define DLL_EXPORT __declspec(dllexport)`

11.15 cputrack/fastjpg.cpp File Reference

```
#include "std_incl.h"
#include <cstdio>
#include "jpeglib.h"
#include "utils.h"
#include "TeLibJpeg\jmemdstsrc.h"
```

Classes

- struct [my_error_mgr](#)

Functions

- int [ReadJPEGFile](#) (`uchar *srcbuf, int srclen, uchar **data, int *width, int *height`)
- void [WriteJPEGFile](#) (`uchar *data, int w, int h, const char *filename, int quality`)
- void [FloatToJPEGFile](#) (`const char *name, const float *d, int w, int h`)

11.15.1 Function Documentation

11.15.1.1 `void FloatToJPEGFile(const char * name, const float * d, int w, int h)`

```
190 {
191     uchar* zlut_bytes = floatToNormalizedInt(d, w,h, (
192         uchar)255);
193     WriteJPEGFile(zlut_bytes, w, h, name, 99);
194     delete[] zlut_bytes;
```

11.15.1.2 int ReadJPEGFile (uchar * *srcbuf*, int *srclen*, uchar ** *data*, int * *width*, int * *height*)

```

13 {
14     struct jpeg_decompress_struct cinfo;
15
16     JSAMPARRAY buffer;           /* Output row buffer */
17     int row_stride;             /* physical row width in output buffer */
18     my_error_mgr jerr;
19     cinfo.err = jpeg_std_error(&jerr.pub);
20     jpeg_create_decompress(&cinfo);
21
22     j_mem_src(&cinfo, srcbuf, srclen);
23
24     /* Step 3: read file parameters with jpeg_read_header() */
25     jpeg_read_header(&cinfo, TRUE);
26
27     jpeg_start_decompress(&cinfo);
28     row_stride = cinfo.output_width * cinfo.output_components;
29     /* Make a one-row-high sample array that will go away when done with image */
30     buffer = (*cinfo.mem->alloc_sarray) ((j_common_ptr) &cinfo, JPOOL_IMAGE, row_stride, 1);
31
32     *width = cinfo.output_width;
33     *height = cinfo.output_height;
34     *data = new uchar[cinfo.output_width*cinfo.output_height];
35
36 //  ResizeLVAArray2D(output, cinfo.output_height, cinfo.output_width);
37
38     /* Step 6: while (scan lines remain to be read) */
39     /*          jpeg_read_scanlines(...); */
40
41     /* Here we use the library's state variable cinfo.output_scanline as the
42      * loop counter, so that we don't have to keep track ourselves.
43      */
44     uchar* dst = *data;
45     while (cinfo.output_scanline < cinfo.output_height) {
46         /* jpeg_read_scanlines expects an array of pointers to scanlines.
47          * Here the array is only one element long, but you could ask for
48          * more than one scanline at a time if that's more convenient.
49         */
50         jpeg_read_scanlines(&cinfo, buffer, 1);
51         /* Assume put_scanline_somewhere wants a pointer and sample count. */
52
53         unsigned char* src = buffer[0];
54         if (cinfo.output_components == 1) {
55             memcpy(dst, src, cinfo.output_width);
56         } else {
57             for (uint x=0;x<cinfo.output_width;x++)
58                 dst[x] = src[x * cinfo.output_components];
59         }
60         dst += cinfo.output_width;
61     }
62
63     /* Step 7: Finish decompression */
64     jpeg_finish_decompress(&cinfo);
65     /* We can ignore the return value since suspension is not possible
66      * with the stdio data source.
67      */
68
69     /* Step 8: Release JPEG decompression object */
70
71     /* This is an important step since it will release a good deal of memory. */
72     jpeg_destroy_decompress(&cinfo);
73
74     /* After finish_decompress, we can close the input file.
75      * Here we postpone it until after no more JPEG errors are possible,
76      * so as to simplify the setjmp error logic above. (Actually, I don't
77      * think that jpeg_destroy can do an error exit, but why assume anything...)
78      */
79 //   fclose(infile);
80
81     /* At this point you may want to check to see whether any corrupt-data
82      * warnings occurred (test whether jerr.pub.num_warnings is nonzero).
83      */
84
85     return 1;
86 }
```

11.15.1.3 void WriteJPEGFile (uchar * *data*, int *w*, int *h*, const char * *filename*, int *quality*)

```

90 {
91     /* This struct contains the JPEG compression parameters and pointers to
```

```

92  * working space (which is allocated as needed by the JPEG library).
93  * It is possible to have several such structures, representing multiple
94  * compression/decompression processes, in existence at once. We refer
95  * to any one struct (and its associated working data) as a "JPEG object".
96  */
97 struct jpeg_compress_struct cinfo;
98 /* This struct represents a JPEG error handler. It is declared separately
99  * because applications often want to supply a specialized error handler
100 * (see the second half of this file for an example). But here we just
101 * take the easy way out and use the standard error handler, which will
102 * print a message on stderr and call exit() if compression fails.
103 * Note that this struct must live as long as the main JPEG parameter
104 * struct, to avoid dangling-pointer problems.
105 */
106 struct jpeg_error_mgr jerr;
107 /* More stuff */
108 JSAMPROW row_pointer[1]; /* pointer to JSAMPLE row[s] */
109 int row_stride; /* physical row width in image buffer */
110
111 /* Step 1: allocate and initialize JPEG compression object */
112
113 /* We have to set up the error handler first, in case the initialization
114 * step fails. (Unlikely, but it could happen if you are out of memory.)
115 * This routine fills in the contents of struct jerr, and returns jerr's
116 * address which we place into the link field in cinfo.
117 */
118 cinfo.err = jpeg_std_error(&jerr);
119 /* Now we can initialize the JPEG compression object. */
120 jpeg_create_compress(&cinfo);
121
122 /* Step 2: specify data destination (eg, a file)
123 /* Note: steps 2 and 3 can be done in either order. */
124
125 uint len = w*h * 2;
126 uchar* memBuf = new uchar[len];
127 j_mem_dest(&cinfo, (void**)&memBuf, &len);
128 //jpeg_stdio_dest(&cinfo, outfile);
129
130 /* Step 3: set parameters for compression */
131
132 /* First we supply a description of the input image.
133  * Four fields of the cinfo struct must be filled in:
134  */
135 cinfo.image_width = w; /* image width and height, in pixels */
136 cinfo.image_height = h;
137 cinfo.input_components = 1; /* # of color components per pixel */
138 cinfo.in_color_space = JCS_GRAYSCALE; /* colorspace of input image */
139 /* Now use the library's routine to set default compression parameters.
140  * (You must set at least cinfo.in_color_space before calling this,
141  * since the defaults depend on the source color space.)
142  */
143 jpeg_set_defaults(&cinfo);
144 /* Now you can set any non-default parameters you wish to.
145  * Here we just illustrate the use of quality (quantization table) scaling:
146  */
147 jpeg_set_quality(&cinfo, quality, TRUE /* limit to baseline-JPEG values */);
148
149 /* Step 4: Start compressor */
150
151 /* TRUE ensures that we will write a complete interchange-JPEG file.
152  * Pass TRUE unless you are very sure of what you're doing.
153  */
154 jpeg_start_compress(&cinfo, TRUE);
155
156 /* Step 5: while (scan lines remain to be written) */
157 /*      jpeg_write_scanlines(...); */
158
159 /* Here we use the library's state variable cinfo.next_scanline as the
160 * loop counter, so that we don't have to keep track ourselves.
161 * To keep things simple, we pass one scanline per call; you can pass
162 * more if you wish, though.
163 */
164 row_stride = w; /* JSAMPLEs per row in image_buffer */
165
166 while (cinfo.next_scanline < cinfo.image_height) {
167  /* jpeg_write_scanlines expects an array of pointers to scanlines.
168   * Here the array is only one element long, but you could pass
169   * more than one scanline at a time if that's more convenient.
170   */
171  row_pointer[0] = &data[cinfo.next_scanline * row_stride];
172  (void) jpeg_write_scanlines(&cinfo, row_pointer, 1);
173 }
174 jpeg_finish_compress(&cinfo);
175 jpeg_destroy_compress(&cinfo);
176
177 FILE *f = fopen(filename, "wb");

```

```

179  if (f) {
180      fwrite(memBuf, 1, len, f);
181      fclose(f);
182  } else
183      dbgprintf("Failed to open file %s for writing\n", filename);
184
185  delete[] memBuf;
186 }

```

11.16 cputrack/FFT2DTracker.h File Reference

```
#include "scalar_types.h"
```

Classes

- class [FFT2DTracker](#)

11.17 cputrack/FisherMatrix.h File Reference

```

#include <functional>
#include "LsqQuadraticFit.h"
#include "threads.h"
#include "random_distr.h"

```

Classes

- class [SampleFisherMatrix](#)

11.18 cputrack/hash_templates.h File Reference

```
#include <hash_map>
#include <hash_set>
```

Classes

- class [qtrk::hash_map< TKey, T >](#)
- class [qtrk::hash_set< T >](#)

Namespaces

- [qtrk](#)

11.19 cputrack/kissfft.h File Reference

```
#include "utils.h"
```

Classes

- struct `kissfft_utils::traits< T_scalar >`
- class `kissfft< T_Scalar, T_traits >`

Namespaces

- `kissfft_utils`

11.20 cputrack/labview.h File Reference

```
#include "extcode.h"
#include "niimaq.h"
#include <complex>
#include "lv_prolog.h"
#include "lv_epilog.h"
```

Classes

- struct `ErrorCluster`
- struct `LVArray< T >`
- struct `LVArray2D< T >`
- struct `LVArray3D< T >`
- struct `LVArrayND< T, N >`
- struct `LVDataType< T >`
- struct `LVDataType< float >`
- struct `LVDataType< double >`
- struct `LVDataType< int8_t >`
- struct `LVDataType< int16_t >`
- struct `LVDataType< int32_t >`
- struct `LVDataType< int64_t >`
- struct `LVDataType< uint8_t >`
- struct `LVDataType< uint16_t >`
- struct `LVDataType< uint32_t >`
- struct `LVDataType< uint64_t >`
- struct `LVDataType< std::complex< float > >`
- struct `LVDataType< std::complex< double > >`

Macros

- `#define _STDINT_H`

Typedefs

- `typedef LVArray< float > ** ppFloatArray`
- `typedef LVArray2D< float > ** ppFloatArray2`

Functions

- `template<typename T >`
`void ResizeLVArray2D (LVArray2D< T > **&d, int rows, int cols)`
- `template<typename T >`
`void ResizeLVArray3D (LVArray3D< T > **&d, int depth, int rows, int cols)`
- `template<typename T , int N>`
`void ResizeLVArray (LVArrayND< T, N > **&d, int *dims)`
- `template<typename T >`
`void ResizeLVArray (LVArray< T > **&d, int elems)`
- `void ArgumentErrorMsg (ErrorCluster *e, const std::string &msg)`
- `void SetLVString (LStrHandle str, const char *text)`
- `MgErr FillErrorCluster (MgErr err, const char *message, ErrorCluster *error)`
- `std::vector< std::string > LVGetStringArray (int count, LStrHandle *str)`
- `bool ValidateTracker (QueuedTracker *tracker, ErrorCluster *e, const char *funcname)`

11.20.1 Macro Definition Documentation

11.20.1.1 `#define _STDINT_H`

11.21 cputrack/LsqQuadraticFit.h File Reference

Classes

- class `LsqSqQuadFit< T >`
- struct `LsqSqQuadFit< T >::Coeff`
- class `ComputeMaxInterp< T, numPts >`

Macros

- `#define CUDA_SUPPORTED_FUNC`

11.21.1 Macro Definition Documentation

11.21.1.1 `#define CUDA_SUPPORTED_FUNC`

11.22 cputrack/lv_cputrack_api.cpp File Reference

```
#include "std_incl.h"
#include <Windows.h>
#include "random_distr.h"
#include "labview.h"
#include "cpu_tracker.h"
```

Functions

- `CDLL_EXPORT CPUTracker *DLL_CALLCONV create_tracker (uint w, uint h, uint xcorw)`
- `CDLL_EXPORT void DLL_CALLCONV destroy_tracker (CPUTracker *tracker)`
- `CDLL_EXPORT void DLL_CALLCONV compute_com (CPUTracker *tracker, float *out)`
- `CDLL_EXPORT int DLL_CALLCONV compute_xcor (CPUTracker *tracker, vector2f *position, int iterations, int profileWidth)`
- `CDLL_EXPORT int DLL_CALLCONV compute_qi (CPUTracker *tracker, vector2f *position, int iterations, int radialSteps, int angularStepsPerQ, float minRadius, float maxRadius, LVArray< float > **radialweights)`
- `CDLL_EXPORT void DLL_CALLCONV set_image_from_memory (CPUTracker *tracker, LVArray2D< uchar > **pData, ErrorCluster *error)`
- `CDLL_EXPORT void DLL_CALLCONV set_image_u8 (CPUTracker *tracker, LVArray2D< uchar > **pData, ErrorCluster *error)`
- `CDLL_EXPORT void DLL_CALLCONV set_image_u16 (CPUTracker *tracker, LVArray2D< ushort > **pData, ErrorCluster *error)`
- `CDLL_EXPORT void DLL_CALLCONV set_image_float (CPUTracker *tracker, LVArray2D< float > **pData, ErrorCluster *error)`
- `CDLL_EXPORT float DLL_CALLCONV compute_z (CPUTracker *tracker, float *center, int angularSteps, int zlut_index, uint *error, LVArray< float > **profile, int *bestIndex, LVArray< float > **errorCurve)`
- `CDLL_EXPORT void DLL_CALLCONV get_debug_img_as_array (CPUTracker *tracker, LVArray2D< float > **pdBglImg)`
- `CDLL_EXPORT void DLL_CALLCONV compute_cr (CPUTracker *tracker, LVArray< float > **result, int radialSteps, float *radii, float *center, uint *boundaryHit, LVArray2D< float > **crpmap)`
- `CDLL_EXPORT float DLL_CALLCONV compute_asymmetry (CPUTracker *tracker, LVArray< float > **result, int radialSteps, float *radii, float *center, uint *boundaryHit)`
- `CDLL_EXPORT void DLL_CALLCONV compute_radial_profile (CPUTracker *tracker, LVArray< float > **result, int angularSteps, float *radii, float *center, uint *boundaryHit)`
- `CDLL_EXPORT void DLL_CALLCONV set_ZLUT (CPUTracker *tracker, LVArray3D< float > **pZlut, float *radii, int angular_steps, bool useCorrelation, LVArray< float > **radialweights, bool normalize)`
- `CDLL_EXPORT void DLL_CALLCONV get_ZLUT (CPUTracker *tracker, int zlutIndex, LVArray2D< float > **dst)`
- `CDLL_EXPORT void DLL_CALLCONV generate_test_image (LVArray2D< float > **img, float xp, float yp, float size, float photoncount)`
- `CDLL_EXPORT void DLL_CALLCONV generate_image_from_lut (LVArray2D< float > **image, LVArray2D< float > **lut, float *LUTradii, vector3f *position, float pixel_max, int useSplineInterp, int samplesPerPixel)`

11.22.1 Function Documentation

11.22.1.1 CDLL_EXPORT float DLL_CALLCONV compute_asymmetry (CPUTracker * tracker, LVArray< float > ** result, int radialSteps, float * radii, float * center, uint * boundaryHit)

```

154 {
155     LVArray<float>* dst = *result;
156     bool bhit = false;
157     float asym = tracker->ComputeAsymmetry(*((vector2f*)center, radialSteps, dst->
158     dimSize, radii[0], radii[1], dst->elem));
159     if (boundaryHit) *boundaryHit = bhit ? 1 : 0;
160 }

```

11.22.1.2 CDLL_EXPORT void DLL_CALLCONV compute_com (CPUTracker * tracker, float * out)

```

41 {
42     vector2f com = tracker->ComputeMeanAndCOM();
43     out[0] = com.x;
44     out[1] = com.y;
45 }

```

11.22.1.3 **CDLL_EXPORT void DLL_CALLCONV compute_crp (CPUTracker * tracker, LVArray< float > ** result, int radialSteps, float * radii, float * center, uint * boundaryHit, LVArray2D< float > ** crpmap)**

```
149 {
150 }
```

11.22.1.4 **CDLL_EXPORT int DLL_CALLCONV compute_qi (CPUTracker * tracker, vector2f * position, int iterations, int radialSteps, int angularStepsPerQ, float minRadius, float maxRadius, LVArray< float > ** radialweights)**

```
57 {
58     bool boundaryHit;
59
60     float* rw = 0;
61     if (radialweights && (*radialweights)->dimSize == radialSteps)
62         rw = (*radialweights)->elem;
63
64     *position = tracker->ComputeQI(*position, iterations, radialSteps, angularStepsPerQ, 1,
65     minRadius, maxRadius, boundaryHit, rw);
65     return boundaryHit ? 1 : 0;
66 }
```

11.22.1.5 **CDLL_EXPORT void DLL_CALLCONV compute_radial_profile (CPUTracker * tracker, LVArray< float > ** result, int angularSteps, float * radii, float * center, uint * boundaryHit)**

```
164 {
165     LVArray<float>* dst = *result;
166     bool bhit = false;
167     tracker->ComputeRadialProfile(&dst->elem[0], dst->
168     dimSize, angularSteps, radii[0], radii[1], *(vector2f*)center, false, &bhit);
169     if (boundaryHit) *boundaryHit = bhit ? 1 : 0;
170 }
```

11.22.1.6 **CDLL_EXPORT int DLL_CALLCONV compute_xcor (CPUTracker * tracker, vector2f * position, int iterations, int profileWidth)**

```
48 {
49     bool boundaryHit;
50     *position = tracker->ComputeXCorInterpolated(*position, iterations, profileWidth
51     , boundaryHit);
52     return boundaryHit ? 1 : 0;
53 }
```

11.22.1.7 **CDLL_EXPORT float DLL_CALLCONV compute_z (CPUTracker * tracker, float * center, int angularSteps, int zlut_index, uint * error, LVArray< float > ** profile, int * bestIndex, LVArray< float > ** errorCurve)**

```
113 {
114     bool boundaryHit=false;
115     if (profile)
116         ResizeLVArray(profile, tracker->zlut_res);
117
118     if (errorCurve) {
119         ResizeLVArray(errorCurve, tracker->zlut_planes);
120     }
121
122     int maxPos;
123     float* prof = profile ? (*profile)->elem : ALLOCA_ARRAY(float, tracker->
124     zlut_res);
124     tracker->ComputeRadialProfile(prof, tracker->zlut_res, angularSteps, tracker
->zlut_minradius, tracker->zlut_maxradius, *(vector2f*) center, false, &
```

```

125     boundaryHit, true);
126     float z= tracker->LUTProfileCompare(prof, zlut_index, errorCurve ? (*errorCurve)->
127     elem : 0, CPUTracker::LUTProfMaxQuadraticFit, 0, &maxPos);
128
129     if (bestIndex) *bestIndex=maxPos;
130
131     if (error)
132         *error = boundaryHit?1:0;
133
134     return z;
135 }
```

11.22.1.8 CDLL_EXPORT CPUTracker* DLL_CALLCONV create_tracker(uint w, uint h, uint xcorw)

```

16 {
17     try {
18         Sleep(300);
19         return new CPUTracker(w,h,xcorw);
20     }
21     catch(const std::exception& e)
22     {
23         dbgout("Exception: " + std::string(e.what()) + "\n");
24         return 0;
25     }
26 }
```

11.22.1.9 CDLL_EXPORT void DLL_CALLCONV destroy_tracker(CPUTracker * tracker)

```

29 {
30     try {
31         delete tracker;
32     }
33     catch(const std::exception& e)
34     {
35         dbgout("Exception: " + std::string(e.what()) + "\n");
36     }
37 }
```

11.22.1.10 CDLL_EXPORT void DLL_CALLCONV generate_image_from_lut(LVArray2D<float> ** image, LVArray2D<float> ** lut, float * LUTradii, vector3f * position, float pixel_max, int useSplineInterp, int samplesPerPixel)

```

217 {
218     ImageData img((*image)->elem, (*image)->dimSizes[1], (*image)->dimSizes[0]);
219     ImageData zlut((*lut)->elem, (*lut)->dimSizes[1], (*lut)->dimSizes[0]);
220
221     GenerateImageFromLUT(&img, &zlut, LUTradii[0], LUTradii[1], *position,
222     useSplineInterp!=0, samplesPerPixel);
223     if (pixel_max>0) {
224         img.normalize();
225         ApplyPoissonNoise(img, pixel_max);
226     }
227 }
```

11.22.1.11 CDLL_EXPORT void DLL_CALLCONV generate_test_image(LVArray2D<float> ** img, float xp, float yp, float size, float photoncount)

```

201 {
202     try {
203         ImageData data((*img)->elem, (*img)->dimSizes[1],(*img)->dimSizes[0]);
204         GenerateTestImage(data, xp, yp, size, photoncount);
205     }
206     catch(const std::exception& e)
207     {
208         dbgout("Exception: " + std::string(e.what()) + "\n");
209     }
210 }
```

11.22.1.12 CDLL_EXPORT void DLL_CALLCONV get_debug_img_as_array (CPUTracker * *tracker*, LVArray2D< float > ** *pdbgImg*)

```

136 {
137     float* src = tracker->GetDebugImage();
138     if (src) {
139         ResizeLVArray2D(pdbgImg, tracker->GetHeight(), tracker->
140             GetWidth());
141         LVArray2D<float>* dst = *pdbgImg;
142         // dbgout(Sprintf("copying %d elements to Labview array\n", len));
143         for (int i=0;i<dst->numElem();i++)
144             dst->elem[i] = src[i];
145     }
146 }
```

11.22.1.13 CDLL_EXPORT void DLL_CALLCONV get_ZLUT (CPUTracker * *tracker*, int *zlutIndex*, LVArray2D< float > ** *dst*)

```

193 {
194     float* zlut = tracker->GetRadialZLUT(zlutIndex);
195     ResizeLVArray2D(dst, tracker->zlut_planes, tracker->
196         zlut_res);
197     std::copy(zlut, zlut+(tracker->zlut_planes*tracker->zlut_res), (*dst)->elem);
198 }
```

11.22.1.14 CDLL_EXPORT void DLL_CALLCONV set_image_float (CPUTracker * *tracker*, LVArray2D< float > ** *pData*, ErrorCluster * *error*)

```

102 {
103     LVArray2D<float>* data = *pData;
104     if (data->dimSizes[0] != tracker->GetWidth() || data->
105         dimSizes[1] != tracker->GetHeight()) {
106         ArgumentErrorMsg(error, "Given image has invalid dimensions");
107         return;
108     }
109     tracker->SetImageFloat( data->elem );
```

11.22.1.15 CDLL_EXPORT void DLL_CALLCONV set_image_from_memory (CPUTracker * *tracker*, LVArray2D< uchar > ** *pData*, ErrorCluster * *error*)

```

70 {
71     LVArray2D<uchar>* data = *pData;
72     if (data->dimSizes[0] != tracker->GetWidth() || data->
73         dimSizes[1] != tracker->GetHeight()) {
74         ArgumentErrorMsg(error, "Given image has invalid dimensions");
75         return;
76     }
77     tracker->SetImage8Bit( data->elem, tracker->GetWidth() );
```

11.22.1.16 CDLL_EXPORT void DLL_CALLCONV set_image_u16 (CPUTracker * *tracker*, LVArray2D< ushort > ** *pData*, ErrorCluster * *error*)

```

92 {
93     LVArray2D<ushort>* data = *pData;
94     if (data->dimSizes[0] != tracker->GetWidth() || data->
95         dimSizes[1] != tracker->GetHeight()) {
96         ArgumentErrorMsg(error, "Given image has invalid dimensions");
97         return;
98     }
99     tracker->SetImage16Bit( data->elem, tracker->GetWidth() * sizeof(
        ushort) );
```

11.22.1.17 **CDLL_EXPORT void DLL_CALLCONV set_image_u8 (CPUTracker * tracker, LVArray2D< uchar > **
pData, ErrorCluster * error)**

```

82 {
83     LVArray2D<uchar>* data = *pData;
84     if (data->dimSizes[0] != tracker->GetWidth() || data->
85         dimSizes[1] != tracker->GetHeight()) {
86         ArgumentErrorMsg(error, "Given image has invalid dimensions");
87         return;
88     }
89     tracker->SetImage8Bit( data->element, tracker->GetWidth() );

```

11.22.1.18 **CDLL_EXPORT void DLL_CALLCONV set_ZLUT (CPUTracker * tracker, LVArray3D< float > ** pZlut,
float * radii, int angular_steps, bool useCorrelation, LVArray< float > ** radialweights, bool normalize)**

```

176 {
177     LVArray3D<float>* zlut = *pZlut;
178
179     int numLUTs = zlut->dimSizes[0];
180     int planes = zlut->dimSizes[1];
181     int res = zlut->dimSizes[2];
182
183     if (normalize) {
184         NormalizeZLUT( zlut->element, numLUTs, planes, res);
185     }
186
187     tracker->SetRadialZLUT(zlut->element, planes, res, numLUTs, radii[0], radii[1], true,
188     useCorrelation);
189     if (radialweights)
190         tracker->SetRadialWeights( ((*radialweights)->dimSize>0) ? (*radialweights)->element :
190 );
```

11.23 cputrack/lv_qtrk_api.h File Reference

```
#include "std_incl.h"
#include "labview.h"
#include "QueuedTracker.h"
```

Classes

- struct [CUDADeviceInfo](#)

Enumerations

- enum [QueueFrameFlags](#) { QFF_Force32Bit = 0x7fffffff }

Functions

- **CDLL_EXPORT void DLL_CALLCONV qtrk_set_ZLUT** (QueuedTracker *tracker, LVArray3D< float > **pZlut, LVArray< float > **zcmpWindow, int normalize, ErrorCluster *e)
- **CDLL_EXPORT void DLL_CALLCONV qtrk_get_ZLUT** (QueuedTracker *tracker, LVArray3D< float > **pzlut, ErrorCluster *e)
- **CDLL_EXPORT QueuedTracker *DLL_CALLCONV qtrk_create** (QTrkSettings *settings, LStrHandle warnings, ErrorCluster *e)
- **CDLL_EXPORT void DLL_CALLCONV qtrk_destroy** (QueuedTracker *qtrk, ErrorCluster *error)
- **CDLL_EXPORT void DLL_CALLCONV qtrk_queue_u16** (QueuedTracker *qtrk, ErrorCluster *error, LVArray2D< ushort > **data, const LocalizationJob *jobInfo)
- **CDLL_EXPORT void DLL_CALLCONV qtrk_queue_u8** (QueuedTracker *qtrk, ErrorCluster *error, LVArray2D< uchar > **data, const LocalizationJob *jobInfo)
- **CDLL_EXPORT void DLL_CALLCONV qtrk_queue_float** (QueuedTracker *qtrk, ErrorCluster *error, LVArray2D< float > **data, const LocalizationJob *jobInfo)
- **CDLL_EXPORT void DLL_CALLCONV qtrk_queue_pitchedmem** (QueuedTracker *qtrk, uchar *data, int pitch, uint pdt, const LocalizationJob *jobInfo)
- **CDLL_EXPORT void DLL_CALLCONV qtrk_queue_array** (QueuedTracker *qtrk, ErrorCluster *error, LVArray2D< uchar > **data, uint pdt, const LocalizationJob *jobInfo)
- **CDLL_EXPORT uint qtrk_read_timestamp** (uchar *image, int w, int h)
- **CDLL_EXPORT uint DLL_CALLCONV qtrk_queue_frame** (QueuedTracker *qtrk, uchar *image, int pitch, int w, int h, uint pdt, ROIPosition *pos, int numROI, const LocalizationJob *pJobInfo, QueueFrameFlags flags, ErrorCluster *e)
- **CDLL_EXPORT void DLL_CALLCONV qtrk_clear_results** (QueuedTracker *qtrk, ErrorCluster *e)
- **CDLL_EXPORT int DLL_CALLCONV qtrk_resultcount** (QueuedTracker *qtrk, ErrorCluster *e)
- **CDLL_EXPORT void DLL_CALLCONV qtrk_flush** (QueuedTracker *qtrk, ErrorCluster *e)
- **CDLL_EXPORT int DLL_CALLCONV qtrk_get_results** (QueuedTracker *qtrk, LocalizationResult *results, int maxResults, int sortByID, ErrorCluster *e)
- **CDLL_EXPORT int DLL_CALLCONV qtrk_idle** (QueuedTracker *qtrk, ErrorCluster *e)
- **CDLL_EXPORT void DLL_CALLCONV qtrk_generate_image_from_lut** (LVArray2D< float > **image, LVArray2D< float > **lut, float *LUTradii, vector2f *position, float z, float M, float sigma_noise)
- **CDLL_EXPORT void DLL_CALLCONV qtrk_dump_memleaks** ()
- **CDLL_EXPORT void qtrk_get_profile_report** (QueuedTracker *qtrk, LStrHandle str)
- **CDLL_EXPORT int DLL_CALLCONV qtrkcuda_device_count** (ErrorCluster *e)
- **CDLL_EXPORT void DLL_CALLCONV qtrkcuda_set_device_list** (LVArray< int > **devices)

11.24 cputrack/lv_queuetrk_api.cpp File Reference

```
#include "std_incl.h"
#include "utils.h"
#include "labview.h"
#include "QueuedTracker.h"
#include "threads.h"
#include "lv_qtrk_api.h"
#include "ResultManager.h"
#include "FisherMatrix.h"
#include "BeadFinder.h"
```

Functions

- `CDLL_EXPORT void DLL_CALLCONV qtrk_free_all ()`
- `void SetLVString (LStrHandle str, const char *text)`
- `std::vector< std::string > LVGetStringArray (int count, LStrHandle *str)`
- `MgErr FillErrorCluster (MgErr err, const char *message, ErrorCluster *error)`
- `void ArgumentErrorMsg (ErrorCluster *e, const std::string &msg)`
- `bool ValidateTracker (QueuedTracker *tracker, ErrorCluster *e, const char *funcname)`
- `CDLL_EXPORT int qtrk_get_debug_image (QueuedTracker *qtrk, int id, LVArray2D< float > **data, ErrorCluster *e)`
- `CDLL_EXPORT void DLL_CALLCONV qtrk_set_logfile_path (const char *path)`
- `CDLL_EXPORT void DLL_CALLCONV qtrk_get_computed_config (QueuedTracker *qtrk, QTrkComputedConfig *cc, ErrorCluster *err)`
- `CDLL_EXPORT void DLL_CALLCONV qtrk_set_ZLUT (QueuedTracker *tracker, LVArray3D< float > **pzlut, LVArray< float > **zcmpWindow, int normalize, ErrorCluster *e)`
- `CDLL_EXPORT void DLL_CALLCONV qtrk_get_ZLUT (QueuedTracker *tracker, LVArray3D< float > **pzlut, ErrorCluster *e)`
- `CDLL_EXPORT void DLL_CALLCONV qtrk_get_image_lut (QueuedTracker *qtrk, LVArrayND< float, 4 > **imageLUT, ErrorCluster *e)`
- `CDLL_EXPORT void DLL_CALLCONV qtrk_set_image_lut (QueuedTracker *qtrk, LVArrayND< float, 4 > **imageLUT, LVArray3D< float > **radialZLUT, ErrorCluster *e)`
- `CDLL_EXPORT void DLL_CALLCONV qtrk_set_pixel_calib_factors (QueuedTracker *qtrk, float offsetFactor, float gainFactor, ErrorCluster *e)`
- `CDLL_EXPORT void DLL_CALLCONV qtrk_set_pixel_calib (QueuedTracker *qtrk, LVArray3D< float > **offset, LVArray3D< float > **gain, ErrorCluster *e)`
- `CDLL_EXPORT QueuedTracker * qtrk_create (QTrkSettings *settings, LStrHandle warnings, ErrorCluster *e)`
- `CDLL_EXPORT void qtrk_destroy (QueuedTracker *qtrk, ErrorCluster *error)`
- template<typename T >
 `bool CheckImageInput (QueuedTracker *qtrk, LVArray2D< T > **data, ErrorCluster *error)`
- `CDLL_EXPORT void qtrk_queue_u16 (QueuedTracker *qtrk, ErrorCluster *error, LVArray2D< ushort > **data, const LocalizationJob *jobInfo)`
- `CDLL_EXPORT void qtrk_queue_u8 (QueuedTracker *qtrk, ErrorCluster *error, LVArray2D< uchar > **data, const LocalizationJob *jobInfo)`
- `CDLL_EXPORT void qtrk_queue_float (QueuedTracker *qtrk, ErrorCluster *error, LVArray2D< float > **data, const LocalizationJob *jobInfo)`
- `CDLL_EXPORT void qtrk_queue_pitchedmem (QueuedTracker *qtrk, uchar *data, int pitch, uint pdt, const LocalizationJob *jobInfo)`
- `CDLL_EXPORT void qtrk_queue_array (QueuedTracker *qtrk, ErrorCluster *error, LVArray2D< uchar > **data, uint pdt, const LocalizationJob *jobInfo)`
- `CDLL_EXPORT uint qtrk_read_timestamp (uchar *image, int w, int h)`
- `CDLL_EXPORT uint qtrk_queue_frame (QueuedTracker *qtrk, uchar *image, int pitch, int w, int h, uint pdt, ROIPosition *pos, int numROI, const LocalizationJob *pJobInfo, QueueFrameFlags flags, ErrorCluster *e)`
- `CDLL_EXPORT void qtrk_clear_results (QueuedTracker *qtrk, ErrorCluster *e)`
- `CDLL_EXPORT void qtrk_build_lut_plane (QueuedTracker *qtrk, LVArray3D< float > **data, uint flags, int plane, ErrorCluster *err)`
- `CDLL_EXPORT void qtrk_finalize_lut (QueuedTracker *qtrk, ErrorCluster *e)`
- `CDLL_EXPORT int qtrk_get_queue_len (QueuedTracker *qtrk, int *maxQueueLen, ErrorCluster *e)`
- `CDLL_EXPORT int qtrk_resultcount (QueuedTracker *qtrk, ErrorCluster *e)`
- `CDLL_EXPORT void qtrk_flush (QueuedTracker *qtrk, ErrorCluster *e)`
- `CDLL_EXPORT int qtrk_get_results (QueuedTracker *qtrk, LocalizationResult *results, int maxResults, int sortByID, ErrorCluster *e)`
- `CDLL_EXPORT void qtrk_get_zlut_cmpprof (QueuedTracker *qtrk, LVArray2D< float > **output, ErrorCluster *e)`
- `CDLL_EXPORT void qtrk_enable_zlut_cmpprof (QueuedTracker *qtrk, bool enable, ErrorCluster *e)`
- `CDLL_EXPORT void qtrk_set_localization_mode (QueuedTracker *qtrk, uint locType, ErrorCluster *e)`

- **CDLL_EXPORT** int **qtrk_idle** (**QueuedTracker** ***qtrk**, **ErrorCluster** ***e**)
- **CDLL_EXPORT** void **qtrk_compute_zlut_bias_table** (**QueuedTracker** ***qtrk**, int **bias_planes**, **LVArray2D< float >** ****lvresult**, int **smpPerPixel**, int **useSplineInterp**, **ErrorCluster** ***e**)
- **CDLL_EXPORT** void **qtrk_set_zlut_bias_table** (**QueuedTracker** ***qtrk**, **LVArray2D< float >** ****biastbl**, **ErrorCluster** ***e**)
- **CDLL_EXPORT** void **DLL_CALLCONV qtrk_generate_gaussian_spot** (**LVArray2D< float >** ****image**, **vector2f** ***pos**, float **sigma**, float **I0**, float **lbg**, int **applyNoise**)
- **CDLL_EXPORT** void **DLL_CALLCONV qtrk_generate_image_from_lut** (**LVArray2D< float >** ****image**, **LVArray2D< float >** ****lut**, float ***LUTradii**, **vector2f** ***position**, float **z**, float **M**, float **sigma_noise**)
- **CDLL_EXPORT** void **qtrk_dump_memleaks** ()
- **CDLL_EXPORT** void **qtrk_get_profile_report** (**QueuedTracker** ***qtrk**, **LStrHandle** **str**)
- **CDLL_EXPORT** void **qtrk_compute_fisher** (**LVArray2D< float >** ****lut**, **QTrkSettings** ***cfg**, **vector3f** ***pos**, **LVArray2D< float >** ****fisherMatrix**, **LVArray2D< float >** ****inverseMatrix**, **vector3f** ***xyzVariance**, int **Nsamples**, float **maxPixelValue**)
- **CDLL_EXPORT** void **qtrk_find_beads** (uint8_t ***image**, int **pitch**, int **w**, int **h**, int ***smpCornerPos**, int **roi**, float **imgRelDist**, float **acceptance**, **LVArray2D< uint32_t >** ****output**)
- **CDLL_EXPORT** void **test_array_passing** (int **n**, **LVArray< float >** ****flt1D**, **LVArray2D< float >** ****flt2D**, **LVArray< int >** ****int1D**, **LVArray2D< int >** ****int2D**)
- **CDLL_EXPORT** void **qtrk_simulate_tracking** (**QueuedTracker** ***qtrk**, int **nsmp**, int **beadIndex**, **vector3f** ***centerPos**, **vector3f** ***range**, **vector3f** ***outBias**, **vector3f** ***outScatter**, float **photonsPerWell**, **ErrorCluster** ***e**)
- **CDLL_EXPORT** int **qtrkcuda_device_count** (**ErrorCluster** ***e**)
- **CDLL_EXPORT** void **qtrkcuda_get_device** (int **device**, void ***info**, **ErrorCluster** ***e**)

Variables

- static **Threads::Mutex** **trackerListMutex**
- static std::vector< **QueuedTracker** * > **trackerList**

11.24.1 Function Documentation

11.24.1.1 template<typename T > bool CheckImageInput (QueuedTracker * *qtrk*, LVArray2D< T > ** *data*, ErrorCluster * *error*)

```

260 {
261     if (!data) {
262         ArgumentErrorMsg(error, "Image data array is empty");
263         return false;
264     } else if( (*data)->dimSizes[1] != qtrk->cfg.width || (*data)->dimSizes[0] != qtrk->
265     cfg.height ) {
266         ArgumentErrorMsg(error, SPrintf( "Image data array has wrong size (%d,%d).
267             Should be: (%d,%d)", (*data)->dimSizes[1], (*data)->dimSizes[0], qtrk->cfg.
268             width, qtrk->cfg.height));
269         return false;
270     }
271     return true;
272 }
```

11.24.1.2 CDLL_EXPORT void qtrk_build_lut_plane (QueuedTracker * qtrk, LVArray3D< float > ** data, uint flags, int plane, ErrorCluster * err)

```

355 {
356     if (ValidateTracker(qtrk, err, "build_lut_plane")) {
357         if ((*data)->dimSizes[2] != qtrk->cfg.width || (*data)->dimSizes[1] != qtrk->
358             cfg.height) {
359             ArgumentErrorMsg(err, SPrintf("Invalid size: %dx%d. Expecting %dx%d\n",
360             (*data)->dimSizes[2], (*data)->dimSizes[1], qtrk->cfg.width, qtrk->cfg.
361             height));
362             return;
363         }
364         int cnt, planes, rsteps;
365         qtrk->GetRadialZLUTSize(cnt, planes, rsteps);
366         if ((*data)->dimSizes[0] != cnt) {
367             ArgumentErrorMsg(err, SPrintf("Invalid number of images given (%d).
368             Expecting %d", (*data)->dimSizes[0], cnt));
369             return;
370         }
371         qtrk->BuildLUT( (*data)->elem, sizeof(float*) * qtrk->cfg.
372             width, QTrkFloat, plane);
373     }
374 }
```

11.24.1.3 CDLL_EXPORT void qtrk_compute_fisher (LVArray2D< float > ** lut, QTrkSettings * cfg, vector3f * pos, LVArray2D< float > ** fisherMatrix, LVArray2D< float > ** inverseMatrix, vector3f * xyzVariance, int Nsamples, float maxPixelValue)

```

515 {
516     QTrkComputedConfig cc (*cfg);
517     ImageData lutImg( (*lut)->elem, (*lut)->dimSizes[1], (*lut)->dimSizes[0] );
518     SampleFisherMatrix fm(maxPixelValue);
519     Matrix3X3 mat = fm.ComputeAverageFisher(*pos, Nsamples, vector3f(1,1,1),
520     vector3f(1,1,1)*0.001f, cfg->width, cfg->height, [&]
521     ImageData& out, vector3f pos) {
522         GenerateImageFromLUT(&out, &lutImg, cc.zlut_minradius, cc.zlut_maxradius, pos);
523     });
524     if (fisherMatrix) {
525         ResizeLVArray2D( fisherMatrix, 3, 3);
526         for (int i=0;i<9;i++)
527             (*fisherMatrix)->elem[i] = mat[i];
528     }
529     if (inverseMatrix) {
530         Matrix3X3 inv = mat.Inverse();
531         ResizeLVArray2D( inverseMatrix, 3, 3);
532         for (int i=0;i<9;i++)
533             (*inverseMatrix)->elem[i] = inv[i];
534     }
535     if (xyzVariance)
536         *xyzVariance = mat.Inverse().diag();
537 }
538 }
```

11.24.1.4 CDLL_EXPORT void qtrk_compute_zlut_bias_table (QueuedTracker * qtrk, int bias_planes, LVArray2D< float > ** lvresult, int smpPerPixel, int useSplineInterp, ErrorCluster * e)

```

451 {
452     if(ValidateTracker(qtrk, e,"compute_zlut_bias_table")) {
453         CImageData result;
454         qtrk->ComputeZBiasCorrection(bias_planes, &result, smpPerPixel,
455         useSplineInterp!=0);
456         ResizeLVArray2D(lvresult, result.h, result.w);
457         result.copyTo ( (*lvresult)->elem );
458     }
459 }
```

11.24.1.5 CDLL_EXPORT void qtrk_enable_zlut_cmpprof (QueuedTracker * qtrk, bool enable, ErrorCluster * e)

```
430 {
431     if (ValidateTracker(qtrk, e, "enable zlut cmpprof"))
432         qtrk->EnableRadialZLUTCompareProfile(enable);
433 }
```

11.24.1.6 CDLL_EXPORT void qtrk_finalize_lut (QueuedTracker * qtrk, ErrorCluster * e)

```
375 {
376     if (ValidateTracker(qtrk, e, "finalize_lut"))
377         qtrk->FinalizeLUT();
378 }
```

11.24.1.7 CDLL_EXPORT void qtrk_find_beads (uint8_t * image, int pitch, int w, int h, int * smpCornerPos, int roi, float imgRelDist, float acceptance, LVArray2D< uint32_t > ** output)

```
542 {
543     BeadFinder::Config cfg;
544     cfg.img_distance = imgRelDist;
545     cfg.roi = roi;
546     cfg.similarity = acceptance;
547     auto results = BeadFinder::Find(image, pitch, w, h, smpCornerPos[0], smpCornerPos[1], &
548     cfg);
549     ResizeLVArray2D(output, results.size(), 2);
550     for (int i=0;i<results.size();i++)
551     {
552         (*output)->get(i, 0) = results[i].x;
553         (*output)->get(i, 1) = results[i].y;
554     }
555 }
```

11.24.1.8 CDLL_EXPORT void DLL_CALLCONV qtrk_free_all ()

```
20 {
21     trackerListMutex.lock();
22     DeleteAllElems(trackerList);
23     trackerListMutex.unlock();
24 }
```

11.24.1.9 CDLL_EXPORT void DLL_CALLCONV qtrk_generate_gaussian_spot (LVArray2D< float > ** image, vector2f * pos, float sigma, float I0, float Ibg, int applyNoise)

```
477 {
478     ImageData img((*image)->elem, (*image)->dimSizes[1], (*image)->dimSizes[0]);
479     GenerateGaussianSpotImage(&img, *pos, sigma, I0, Ibg);
480     if (applyNoise)
481         ApplyPoissonNoise(img, 1);
482 }
```

11.24.1.10 CDLL_EXPORT void DLL_CALLCONV qtrk_get_computed_config (QueuedTracker * qtrk, QTrkComputedConfig * cc, ErrorCluster * err)

```
95 {
96     if (ValidateTracker(qtrk, err, "get computed config"))
97         *cc = qtrk->cfg;
98 }
```

11.24.1.11 CDLL_EXPORT int qtrk_get_debug_image (QueuedTracker * *qtrk*, int *id*, LVArray2D< float > ** *data*, ErrorCluster * *e*)

```

75 {
76     int w,h;
77     if (ValidateTracker(qtrk, e, "qtrk_get_debug_image")) {
78         float* img;
79         if (qtrk->GetDebugImage(id, &w,&h, &img)) {
80             ResizeLVArray2D(data, h, w);
81             for (int i=0;i<w*h;i++) (*data)->elem[i]=img[i];
82             delete[] img;
83             return 1;
84         }
85     }
86     return 0;
87 }
```

11.24.1.12 CDLL_EXPORT void DLL_CALLCONV qtrk_get_image_lut (QueuedTracker * *qtrk*, LVArrayND< float, 4 > ** *imageLUT*, ErrorCluster * *e*)

```

152 {
153     if (ValidateTracker(qtrk, e, "get_image_lut")) {
154         int dims[4];
155         qtrk->GetImageZLUTSize(dims);
156         ResizeLVArray(imageLUT, dims);
157         if ( (*imageLUT)->numElem () > 0 ){
158             qtrk->GetImageZLUT( (*imageLUT)->elem );
159         }
160     }
161 }
162 }
```

11.24.1.13 CDLL_EXPORT int qtrk_get_queue_len (QueuedTracker * *qtrk*, int * *maxQueueLen*, ErrorCluster * *e*)

```

382 {
383     if (ValidateTracker(qtrk, e, "fullqueue"))
384         return qtrk->GetQueueLength(maxQueueLen);
385     return 0;
386 }
```

11.24.1.14 CDLL_EXPORT void qtrk_get_zlut_cmpprof (QueuedTracker * *qtrk*, LVArray2D< float > ** *output*, ErrorCluster * *e*)

```

418 {
419     if (ValidateTracker(qtrk, e, "get zlut compare profiles"))
420     {
421         int cnt,planes,rstps;
422         qtrk->GetRadialZLUTSize(cnt,planes,rstps);
423         ResizeLVArray2D(output, cnt, planes);
424         qtrk->GetRadialZLUTCompareProfile((*output)->elem);
425     }
426 }
```

11.24.1.15 CDLL_EXPORT void DLL_CALLCONV qtrk_set_image_lut (QueuedTracker * *qtrk*, LVArrayND< float, 4 > ** *imageLUT*, LVArray3D< float > ** *radialZLUT*, ErrorCluster * *e*)

```

166 {
167     if (ValidateTracker(qtrk, e, "set_image_lut")) {
168         if ( (*imageLUT)->numElem () == 0 ){
169             qtrk->SetImageZLUT ( 0, 0, 0 );
170         }
171         else {
172             qtrk->SetImageZLUT( (*imageLUT)->elem, (*radialZLUT)->elem, (*imageLUT)->dimSizes )
173         }
174     }
175 }
```

11.24.1.16 CDLL_EXPORT void qtrk_set_localization_mode (QueuedTracker * qtrk, uint locType, ErrorCluster * e)

```
437 {
438     if (ValidateTracker(qtrk, e, "set_localization_mode")) {
439         qtrk->SetLocalizationMode( (LocMode_t)locType );
440     }
441 }
```

11.24.1.17 CDLL_EXPORT void DLL_CALLCONV qtrk_set_logfile_path (const char * path)

```
90 {
91     dbgsetlogfile(path);
92 }
```

11.24.1.18 CDLL_EXPORT void DLL_CALLCONV qtrk_set_pixel_calib (QueuedTracker * qtrk, LVArray3D< float > ** offset, LVArray3D< float > ** gain, ErrorCluster * e)

```
185 {
186     if (ValidateTracker(qtrk, e, "set pixel calibration images")) {
187         int count, planes, radialSteps;
188         qtrk->GetRadialZLUTSize(count, planes, radialSteps);
189
190         float *offset_data = 0, *gain_data = 0;
191
192         if ((*offset)->dimSizes[0] != 0) {
193             if (qtrk->cfg.width != (*offset)->dimSizes[2] || qtrk->cfg.height != (*offset)->dimSizes[1]) {
194                 ArgumentErrorMsg(e, Sprintf("set_pixel_calib: Offset images passed with invalid dimension (%d,%d)", (*offset)->dimSizes[2], (*offset)->dimSizes[1]));
195                 return;
196             }
197             if (count != (*offset)->dimSizes[0]) {
198                 ArgumentErrorMsg(e, Sprintf("set_pixel_calib: Expecting offset to have %d images (same as ZLUT). %d given", count, (*offset)->dimSizes[0]));
199                 return;
200             }
201             offset_data = (*offset)->elem;
202         }
203
204         if ((*gain)->dimSizes[0] != 0) {
205             if (qtrk->cfg.width != (*gain)->dimSizes[2] || qtrk->cfg.height != (*gain)->dimSizes[1]) {
206                 ArgumentErrorMsg(e, Sprintf("set_pixel_calib: Gain images passed with invalid dimension (%d,%d)", (*gain)->dimSizes[2], (*gain)->dimSizes[1]));
207                 return;
208             }
209             if (count != (*gain)->dimSizes[0]) {
210                 ArgumentErrorMsg(e, Sprintf("set_pixel_calib: Expecting gain to have %d images (same as ZLUT). %d given", count, (*gain)->dimSizes[0]));
211                 return;
212             }
213             gain_data = (*gain)->elem;
214         }
215
216         qtrk->SetPixelCalibrationImages(offset_data, gain_data);
217     }
218 }
```

11.24.1.19 CDLL_EXPORT void DLL_CALLCONV qtrk_set_pixel_calib_factors (QueuedTracker * qtrk, float offsetFactor, float gainFactor, ErrorCluster * e)

```
178 {
179     if (ValidateTracker(qtrk, e, "set pixel calib factors")) {
180         qtrk->SetPixelCalibrationFactors(offsetFactor, gainFactor);
181     }
182 }
```

11.24.1.20 CDLL_EXPORT void qtrk_set_zlut_bias_table (QueuedTracker * qtrk, LVArray2D< float > ** biastbl, ErrorCluster * e)

```

462 {
463     if (ValidateTracker(qtrk, e, "set zlut bias table")) {
464         int numbeads, planes, radialsteps;
465         qtrk->GetRadialZLUTSize(numbeads, planes, radialsteps);
466
467         if ((*biastbl)->dimSizes[1] != numbeads) {
468             ArgumentErrorMsg(e, SPrintf( "Bias table should be [numbeads] high and
469             [biasplanes] wide. Expected #beads=%d", numbeads) );
470         }
471         else {
472             qtrk->SetZLUTBiasCorrection( ImageData( (*biastbl)->elem,
473             (*biastbl)->dimSizes[0], (*biastbl)->dimSizes[1] ) );
472         }
473     }
474 }
```

11.24.1.21 CDLL_EXPORT void qtrk_simulate_tracking (QueuedTracker * qtrk, int nsmp, int beadIndex, vector3f * centerPos, vector3f * range, vector3f * outBias, vector3f * outScatter, float photonsPerWell, ErrorCluster * e)

```

584 {
585     if (ValidateTracker(qtrk, e, "qtrk_simulate_tracking")) {
586
587         int nZLUT, nPlanes, nRadialSteps;
588         qtrk->GetRadialZLUTSize(nZLUT, nPlanes, nRadialSteps);
589
590         float* lut = new float[nZLUT*nPlanes*nRadialSteps];
591         qtrk->GetRadialZLUT(lut);
592
593         ImageData img = ImageData::alloc(qtrk->cfg.
594             width, qtrk->cfg.height);
595         ImageData zlut;
596         zlut.data = & lut [nRadialSteps*nPlanes*beadIndex];
597         zlut.w = nRadialSteps;
598         zlut.h = nPlanes;
599
600         // Generate images
601         std::vector<vector3f> positions(nsmp);
602         for (int i=0;i<nsmp;i++) {
603             vector3f pos = *centerPos + *range * vector3f(rand_uniform<float>(),
604                 rand_uniform<float>(), rand_uniform<float>());
605             positions[i]=pos;
606             GenerateImageFromLUT( &img, &zlut, qtrk->cfg.
607                 zlut_minradius, qtrk->cfg.zlut_maxradius, pos);
608             ApplyPoissonNoise(img, photonsPerWell);
609             qtrk->ScheduleLocalization((uchar*)img.data, sizeof(float)*img.
610                 w, QTrkFloat,i,i,0,beadIndex);
611         }
612
613         img.free();
614
615         qtrk->Flush();
616         while (!qtrk->IsIdle());
617
618         LocalizationResult* results=new LocalizationResult[nsmp];
619         qtrk->FetchResults(results, nsmp);
620         vector3f sumBias, sumScatter;
621         for (int i=0;i<nsmp;i++) {
622             vector3f truepos = positions [ results[i].job.frame ];
623             vector3f diff = results[i].pos - truepos;
624             sumBias += diff;
625         }
626         vector3f meanBias = sumBias / nsmp;
627         for (int i=0;i<nsmp;i++) {
628             vector3f truepos = positions [ results[i].job.frame ];
629             vector3f diff = results[i].pos - truepos;
630             diff -= meanBias;
631             sumScatter += diff*diff;
632         }
633     }
634 }
```

11.24.1.22 CDLL_EXPORT void qtrkcuda_get_device (int device, void * info, ErrorCluster * e)

```
685 { }
```

11.24.1.23 CDLL_EXPORT void test_array_passing (int n, LVArray< float > ** flt1D, LVArray2D< float > ** flt2D, LVArray< int > ** int1D, LVArray2D< int > ** int2D)

```
559 {
560     for (int i=0;i<(*int2D)->dimSizes[0];i++)
561     {
562         for (int j=0;j<(*int2D)->dimSizes[1];j++)
563         {
564             dbgprintf("%d\t", (*int2D)->get(i, j));
565         }
566         dbgprintf("\n");
567     }
568
569     ResizeLVArray(flt1D, n);
570     ResizeLVArray(int1D, n);
571     ResizeLVArray2D(flt2D, n/2,n);
572     ResizeLVArray2D(int2D, n/2,n);
573     for (int i=0;i<n;i++) {
574         (*int1D)->elem[i]=(i+1)*i;
575         (*flt1D)->elem[i]=sqrtf(i);
576         for (int j=0;j<n/2;j++) {
577             (*int2D)->get(j, i) = j*2+i;
578             (*flt2D)->get(j, i) = j*2+i;
579         }
580     }
581 }
```

11.24.2 Variable Documentation

11.24.2.1 std::vector<QueuedTracker*> trackerList [static]

11.24.2.2 Threads::Mutex trackerListMutex [static]

11.25 cputrack/lv_resultmanager_api.cpp File Reference

```
#include "std_incl.h"
#include "labview.h"
#include "ResultManager.h"
#include "utils.h"
#include <unordered_set>
```

Functions

- static bool ValidRM (ResultManager *rm, ErrorCluster *err)
- CDLL_EXPORT void DLL_CALLCONV rm_destroy_all ()
- CDLL_EXPORT ResultManager *DLL_CALLCONV rm_create (const char *file, const char *frameinfo, ResultManagerConfig *cfg, LStrHandle *names)
- CDLL_EXPORT void DLL_CALLCONV rm_set_tracker (ResultManager *rm, QueuedTracker *qtrk, ErrorCluster *err)
- CDLL_EXPORT void DLL_CALLCONV rm_destroy (ResultManager *rm, ErrorCluster *err)
- CDLL_EXPORT void DLL_CALLCONV rm_store_frame_info (ResultManager *rm, int frame, double timestamp, float *cols, ErrorCluster *err)

- **CDLL_EXPORT** int **DLL_CALLCONV rm_getbeadresults** (**ResultManager** *rm, int start, int numFrames, int bead, **LocalizationResult** *results, **ErrorCluster** *err)
- **CDLL_EXPORT** void **DLL_CALLCONV rm_getframecounters** (**ResultManager** *rm, int *startFrame, int *lastSaveFrame, int *endFrame, int *capturedFrames, int *localizationsDone, int *lostFrames, **ErrorCluster** *err)
- **CDLL_EXPORT** void **DLL_CALLCONV rm_flush** (**ResultManager** *rm, **ErrorCluster** *err)
- **CDLL_EXPORT** int **DLL_CALLCONV rm_getresults** (**ResultManager** *rm, int startFrame, int numFrames, **LocalizationResult** *results, **ErrorCluster** *err)
- **CDLL_EXPORT** void **DLL_CALLCONV rm_removebead** (**ResultManager** *rm, int bead, **ErrorCluster** *err)
- **CDLL_EXPORT** void **DLL_CALLCONV rm_getconfig** (**ResultManager** *rm, **ResultManagerConfig** *cfg, **ErrorCluster** *err)

Variables

- static std::unordered_set< **ResultManager** * > **rm_instances**

11.25.1 Function Documentation

11.25.1.1 **CDLL_EXPORT ResultManager* DLL_CALLCONV rm_create (const char * file, const char * frameinfo, ResultManagerConfig * cfg, LStrHandle * names)**

```

27 {
28     std::vector<std::string> colNames;
29
30     if (names) colNames = LVGetStringArray(cfg->
31         numFrameInfoColumns, names);
32     ResultManager* rm = new ResultManager(file, frameinfo, cfg, colNames);
33     rm_instances.insert(rm);
34     return rm;
35 }
```

11.25.1.2 **CDLL_EXPORT void DLL_CALLCONV rm_destroy (ResultManager * rm, ErrorCluster * err)**

```

45 {
46     if (ValidRM(rm, err)) {
47         rm_instances.erase(rm);
48         delete rm;
49     }
50 }
```

11.25.1.3 **CDLL_EXPORT void DLL_CALLCONV rm_destroy_all ()**

```

22 {
23     DeleteAllElems(rm_instances);
24 }
```

11.25.1.4 **CDLL_EXPORT void DLL_CALLCONV rm_flush (ResultManager * rm, ErrorCluster * err)**

```

92 {
93     if (ValidRM(rm, err)) {
94         rm->Flush();
95     }
96 }
```

11.25.1.5 CDLL_EXPORT int DLL_CALLCONV rm_getbeadresults (ResultManager * rm, int start, int numFrames, int bead, LocalizationResult * results, ErrorCluster * err)

```

63 {
64     if (ValidRM(rm, err)) {
65         if (bead < 0 || bead >= rm->Config().numBeads)
66             ArgumentErrorMsg(err, Sprintf( "Invalid bead index: %d. Accepted range:
67                                     [0-%d]", bead, rm->Config().numBeads));
68         else
69             return rm->GetBeadPositions(start, start+numFrames, bead, results);
70     }
71     return 0;

```

11.25.1.6 CDLL_EXPORT void DLL_CALLCONV rm_getconfig (ResultManager * rm, ResultManagerConfig * cfg, ErrorCluster * err)

```

120 {
121     if (ValidRM(rm, err)) {
122         *cfg = rm->Config();
123     }
124 }

```

11.25.1.7 CDLL_EXPORT void DLL_CALLCONV rm_getframecounters (ResultManager * rm, int * startFrame, int * lastSaveFrame, int * endFrame, int * capturedFrames, int * localizationsDone, int * lostFrames, ErrorCluster * err)

```

76 {
77     if (ValidRM(rm, err)) {
78
79         auto r = rm->GetFrameCounters();
80
81         if (startFrame) *startFrame = r.startFrame;
82         if (lastSaveFrame) *lastSaveFrame = r.lastSaveFrame;
83         if (endFrame) *endFrame = r.processedFrames;
84         if (capturedFrames) *capturedFrames = r.capturedFrames;
85         if (localizationsDone) *localizationsDone = r.localizationsDone;
86         if (lostFrames) *lostFrames = r.lostFrames;
87     }
88 }

```

11.25.1.8 CDLL_EXPORT int DLL_CALLCONV rm_getresults (ResultManager * rm, int startFrame, int numFrames, LocalizationResult * results, ErrorCluster * err)

```

99 {
100     if (ValidRM(rm, err)) {
101         return rm->GetResults(results, startFrame, numFrames);
102     }
103     return 0;
104 }

```

11.25.1.9 CDLL_EXPORT void DLL_CALLCONV rm_removebead (ResultManager * rm, int bead, ErrorCluster * err)

```

107 {
108     if (ValidRM(rm, err)) {
109         if (rm->GetTracker()) {
110             ArgumentErrorMsg(err, "Cannot discard bead results while tracking in progress")
111         }
112         rm->RemoveBeadResults(bead);
113     }
114 }
115 }

```

11.25.1.10 **CDLL_EXPORT void DLL_CALLCONV rm_set_tracker (ResultManager * rm, QueuedTracker * qtrk, ErrorCluster * err)**

```
38 {
39     if (ValidRM(rm, err)) {
40         rm->SetTracker(qtrk);
41     }
42 }
```

11.25.1.11 **CDLL_EXPORT void DLL_CALLCONV rm_store_frame_info (ResultManager * rm, int frame, double timestamp, float * cols, ErrorCluster * err)**

```
53 {
54     if (ValidRM(rm, err)) {
55 #ifdef _DEBUG
56         dbgprintf("rm_store_frame_info: frame=%d, ts=%f\n", frame,timestamp);
57 #endif
58         rm->StoreFrameInfo(frame, timestamp, cols);
59     }
60 }
```

11.25.1.12 **static bool ValidRM (ResultManager * rm, ErrorCluster * err) [static]**

```
13 {
14     if(rm_instances.find(rm) == rm_instances.end()) {
15         ArgumentErrorMsg(err, "Invalid ResultManager instance passed.");
16         return false;
17     }
18     return true;
19 }
```

11.25.2 Variable Documentation

11.25.2.1 **std::unordered_set<ResultManager*> rm_instances [static]**

11.26 cputrack/memdbg.cpp File Reference

11.27 cputrack/memdbg.h File Reference

11.28 cputrack/qtrk_c_api.cpp File Reference

```
#include "std_incl.h"
#include "QueuedTracker.h"
```

Functions

- **CDLL_EXPORT** QueuedTracker ***DLL_CALLCONV** QTrkCreateInstance (QTrkSettings *cfg)
Create a QTrk instance and return a pointer to it.
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkFreeInstance (QueuedTracker *qtrk)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkSetLocalizationMode (QueuedTracker *qtrk, LocMode_t locType)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkScheduleLocalization (QueuedTracker *qtrk, void *data, int pitch, QTRK_PixelDataType pdt, const LocalizationJob *jobInfo)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkClearResults (QueuedTracker *qtrk)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkFlush (QueuedTracker *qtrk)
- **CDLL_EXPORT** int **DLL_CALLCONV** QTrkScheduleFrame (QueuedTracker *qtrk, void *imgptr, int pitch, int width, int height, ROIPosition *positions, int numROI, QTRK_PixelDataType pdt, const LocalizationJob *jobInfo)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkSetRadialZLUT (QueuedTracker *qtrk, float *data, int count, int planes, float *zcmp)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkGetRadialZLUT (QueuedTracker *qtrk, float *dst)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkGetRadialZLUTSize (QueuedTracker *qtrk, int *count, int *planes, int *radialsteps)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkBuildLUT (QueuedTracker *qtrk, void *data, int pitch, QTRK_PixelDataType pdt, bool imageLUT, int plane)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkFinalizeLUT (QueuedTracker *qtrk)
- **CDLL_EXPORT** int **DLL_CALLCONV** QTrkGetResultCount (QueuedTracker *qtrk)
- **CDLL_EXPORT** int **DLL_CALLCONV** QTrkFetchResults (QueuedTracker *qtrk, LocalizationResult *results, int maxResults)
- **CDLL_EXPORT** int **DLL_CALLCONV** QTrkGetQueueLength (QueuedTracker *qtrk, int *maxQueueLen)
- **CDLL_EXPORT** bool **DLL_CALLCONV** QTrkIsIdle (QueuedTracker *qtrk)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkGetProfileReport (QueuedTracker *qtrk, char *dst, int maxStrLen)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkGetWarnings (QueuedTracker *qtrk, char *dst, int maxStrLen)
- **CDLL_EXPORT** void **DLL_CALLCONV** QTrkGetComputedConfig (QueuedTracker *qtrk, QTrkComputedConfig *cfg)

11.29 cputrack/qtrk_c_api.h File Reference

```
#include "dllmacros.h"
```

Classes

- struct **LocalizationJob**
Structure for region of interest metadata.
- struct **LocalizationResult**
Structure for job results.
- struct **QTrkSettings**
*Structure for the settings used by the algorithms implemented in **QueuedTracker**.*
- struct **ROIPosition**
Struct used to define the top-left corner position of an ROI within a frame. ROI is [x .. x+w ; y .. y+h].
- struct **QTrkComputedConfig**
*Structure for derived settings computed from base settings in **QTrkSettings**.*

Macros

- `#define QTrkCUDA_UseList -3`
- `#define QTrkCUDA_UseAll -2`
- `#define QTrkCUDA_UseBest -1`

Typedefs

- `typedef int LocMode_t`

Enumerations

- `enum LocalizeModeEnum { LT_OnlyCOM = 0, LT_XCor1D = 1, LT_QI = 2, LT_Gaussian2D = 4, LT_ZLUTAlign = 8, LT_LocalizeZ = 16, LT_NormalizeProfile = 64, LT_ClearFirstFourPixels = 128, LT_FourierLUT = 256, LT_LocalizeZWeighted = 512, LT_Force32Bit = 0xffffffff }`
Flags for selecting localization type.
- `enum QTRK_PixelDataType { QTrkU8 = 0, QTrkU16 = 1, QTrkFloat = 2 }`
Flags for data type of image data.

Functions

- `CDLL_EXPORT QueuedTracker *DLL_CALLCONV QTrkCreateInstance (QTrkSettings *cfg)`
Create a QTrk instance and return a pointer to it.
- `CDLL_EXPORT void DLL_CALLCONV QTrkFreeInstance (QueuedTracker *qtrk)`
- `CDLL_EXPORT void DLL_CALLCONV QTrkSetLocalizationMode (QueuedTracker *qtrk, LocMode_t locType)`
- `CDLL_EXPORT void DLL_CALLCONV QTrkScheduleLocalization (QueuedTracker *qtrk, void *data, int pitch, QTRK_PixelDataType pdt, const LocalizationJob *jobInfo)`
- `CDLL_EXPORT void DLL_CALLCONV QTrkClearResults (QueuedTracker *qtrk)`
- `CDLL_EXPORT void DLL_CALLCONV QTrkFlush (QueuedTracker *qtrk)`
- `CDLL_EXPORT int DLL_CALLCONV QTrkScheduleFrame (QueuedTracker *qtrk, void *imgptr, int pitch, int width, int height, ROIPosition *positions, int numROI, QTRK_PixelDataType pdt, const LocalizationJob *jobInfo)`
- `CDLL_EXPORT void DLL_CALLCONV QTrkSetRadialZLUT (QueuedTracker *qtrk, float *data, int count, int planes, float *zcmp=0)`
- `CDLL_EXPORT void DLL_CALLCONV QTrkGetRadialZLUT (QueuedTracker *qtrk, float *dst)`
- `CDLL_EXPORT void DLL_CALLCONV QTrkGetRadialZLUTSize (QueuedTracker *qtrk, int *count, int *planes, int *radialsteps)`
- `CDLL_EXPORT void DLL_CALLCONV QTrkBuildLUT (QueuedTracker *qtrk, void *data, int pitch, QTRK_PixelDataType pdt, bool imageLUT, int plane)`
- `CDLL_EXPORT void DLL_CALLCONV QTrkFinalizeLUT (QueuedTracker *qtrk)`
- `CDLL_EXPORT int DLL_CALLCONV QTrkGetResultCount (QueuedTracker *qtrk)`
- `CDLL_EXPORT int DLL_CALLCONV QTrkFetchResults (QueuedTracker *qtrk, LocalizationResult *results, int maxResults)`
- `CDLL_EXPORT int DLL_CALLCONV QTrkGetQueueLength (QueuedTracker *qtrk, int *maxQueueLen)`
- `CDLL_EXPORT bool DLL_CALLCONV QTrkIsIdle (QueuedTracker *qtrk)`
- `CDLL_EXPORT void DLL_CALLCONV QTrkGetProfileReport (QueuedTracker *qtrk, char *dst, int maxStrLen)`
- `CDLL_EXPORT void DLL_CALLCONV QTrkGetWarnings (QueuedTracker *qtrk, char *dst, int maxStrLen)`
- `CDLL_EXPORT void DLL_CALLCONV QTrkGetComputedConfig (QueuedTracker *qtrk, QTrkComputedConfig *cfg)`

11.29.1 Macro Definition Documentation

11.29.1.1 `#define QTrkCUDA_UseAll -2`

11.29.1.2 `#define QTrkCUDA_UseBest -1`

11.29.1.3 `#define QTrkCUDA_UseList -3`

11.29.2 Typedef Documentation

11.29.2.1 `typedef int LocMode_t`

11.29.3 Enumeration Type Documentation

11.29.3.1 `enum LocalizeModeEnum`

Flags for selecting localization type.

Enumerator

- `LT_OnlyCOM` Use only COM.
- `LT_XCor1D` COM+XCor1D.
- `LT_QI` COM+QI.
- `LT_Gaussian2D` 2D Gaussian localization
- `LT_ZLUTAlign` XYZ Alignment with ZLUT.
- `LT_LocalizeZ`
- `LT_NormalizeProfile`
- `LT_ClearFirstFourPixels`
- `LT_FourierLUT`
- `LT_LocalizeZWeighted`
- `LT_Force32Bit`

```

6
7     LT_OnlyCOM = 0,
8     LT_XCor1D = 1,
9     LT_QI = 2,
10    LT_Gaussian2D = 4,
11    LT_ZLUTAlign = 8,
12
13    LT_LocalizeZ = 16,
14    LT_NormalizeProfile = 64,
15    LT_ClearFirstFourPixels = 128,
16    LT_FourierLUT = 256,
17    LT_LocalizeZWeighted = 512,
18
19    LT_Force32Bit = 0xffffffff
20 };

```

11.29.3.2 enum QTRK_PixelDataType

Flags for data type of image data.

Enumerator

QTrkU8

QTrkU16

QTrkFloat

```
26 {
27     QTrkU8 = 0,
28     QTrkU16 = 1,
29     QTrkFloat = 2
30 };
```

11.30 cputrack/QueuedCPUTracker.cpp File Reference

```
#include "std_incl.h"
#include "QueuedCPUTracker.h"
#include <float.h>
#include <functional>
#include "DebugResultCompare.h"
```

Functions

- `QueuedTracker * CreateQueuedTracker (const QTrkComputedConfig &cc)`
- `void SetCUDADevices (int *dev, int ndev)`

11.30.1 Function Documentation

11.30.1.1 `QueuedTracker* CreateQueuedTracker (const QTrkComputedConfig & cc)`

```
9
10     return new QueuedCPUTracker(cc);
11 }
```

11.30.1.2 `void SetCUDADevices (int * dev, int ndev)`

```
12
13 }
```

11.31 cputrack/QueuedCPUTracker.h File Reference

```
#include "threads.h"
#include "QueuedTracker.h"
#include "cpu_tracker.h"
#include <list>
```

Classes

- class [QueuedCPUTracker](#)
CPU implementation of the [QueuedTracker](#) interface.
- struct [QueuedCPUTracker::Thread](#)
- struct [QueuedCPUTracker::Job](#)

11.32 cputrack/QueuedTracker.cpp File Reference

```
#include "std_incl.h"
#include "QueuedTracker.h"
#include "utils.h"
#include "cpu_tracker.h"
```

Macros

- [#define WRITEVAR\(v\) dbgprintf\("Setting: %s set to %g\n", #v, \(float\) v\)](#)
- [#define WRITEVAR\(v\) fprintf_s\(f,"Setting: %s set to %g\n", #v, \(float\) v\)](#)

11.32.1 Macro Definition Documentation

11.32.1.1 [#define WRITEVAR\(v \) dbgprintf\("Setting: %s set to %g\n", #v, \(float\) v\)](#)

11.32.1.2 [#define WRITEVAR\(v \) fprintf_s\(f,"Setting: %s set to %g\n", #v, \(float\) v\)](#)

11.33 cputrack/QueuedTracker.h File Reference

```
#include "std_incl.h"
#include "threads.h"
#include <map>
#include "qtrk_c_api.h"
```

Classes

- struct [TImageData< T >](#)
- class [QueuedTracker](#)
Abstract tracker interface, implemented by [QueuedCUDATracker](#) and [QueuedCPUTracker](#).

Macros

- [#define MIN_RADPROFILE_SMP_COUNT 4](#)
Minimum number of samples for a profile radial bin. Below this the image mean will be used.
- [#define BUILDLUT_IMAGELUT 1](#)
- [#define BUILDLUT_FOURIER 2](#)
- [#define BUILDLUT_NORMALIZE 4](#)
- [#define BUILDLUT_BIASCORRECT 8](#)
- [#define QI_LSQFIT_WEIGHTS { 0.14f, 0.5f, 0.85f, 1.0f, 0.85f, 0.5f, 0.14f }](#)
- [#define QI_LSQFIT_NWEIGHTS 7](#)
- [#define ZLUT_LSQFIT_NWEIGHTS 7](#)
- [#define ZLUT_LSQFIT_WEIGHTS { 0.15f, 0.5f, 0.85f, 1.0f, 0.85f, 0.5f, 0.15f }](#)

Typedefs

- `typedef TImageData< float > ImageData`

Functions

- `void CopyImageToFloat (uchar *data, int width, int height, int pitch, QTRK_PixelDataType pdt, float *dst)`
- `QueuedTracker * CreateQueuedTracker (const QTrkComputedConfig &cc)`
- `void SetCUDADevices (int *devices, int numdev)`
- `int PDT_BytesPerPixel (QTRK_PixelDataType pdt)`

11.33.1 Macro Definition Documentation

11.33.1.1 `#define BUILDLUT_BIASCORRECT 8`

11.33.1.2 `#define BUILDLUT_FOURIER 2`

11.33.1.3 `#define BUILDLUT_IMAGELUT 1`

11.33.1.4 `#define BUILDLUT_NORMALIZE 4`

11.33.1.5 `#define MIN_RADPROFILE_SMP_COUNT 4`

Minimum number of samples for a profile radial bin. Below this the image mean will be used.

11.33.1.6 `#define QI_LSQFIT_NWEIGHTS 7`

11.33.1.7 `#define QI_LSQFIT_WEIGHTS { 0.14f, 0.5f, 0.85f, 1.0f, 0.85f, 0.5f, 0.14f }`

11.33.1.8 `#define ZLUT_LSQFIT_NWEIGHTS 7`

11.33.1.9 `#define ZLUT_LSQFIT_WEIGHTS { 0.15f, 0.5f, 0.85f, 1.0f, 0.85f, 0.5f, 0.15f }`

11.33.2 Typedef Documentation

11.33.2.1 `typedef TImageData<float> ImageData`

11.33.3 Function Documentation

11.33.3.1 `void CopyImageToFloat (uchar * data, int width, int height, int pitch, QTRK_PixelDataType pdt, float * dst)`

```
650 {
651     if (pdt == QTrkU8) {
652         for (int y=0;y<height;y++) {
653             for (int x=0;x<width;x++)
654                 dst[x] = data[x];
655             data += pitch;
656             dst += width;
657     }
}
```

```

658     } else if(pdt == QTrkU16) {
659         for (int y=0;y<height;y++) {
660             ushort* u = (ushort*)data;
661             for (int x=0;x<width;x++)
662                 dst[x] = u[x];
663             data += pitch;
664             dst += width;
665         }
666     } else {
667         for (int y=0;y<height;y++) {
668             float* fsrc = (float*)data;
669             for( int x=0;x<width;x++)
670                 dst[x] = fsrc[x];
671             data += pitch;
672             dst += width;
673         }
674     }
675 }
```

11.33.3.2 QueuedTracker* CreateQueuedTracker (const QTrkComputedConfig & cc)

```

9
10     return new QueuedCPUTracker(cc);
11 }
```

11.33.3.3 int PDT_BytesPerPixel (QTRK_PixelDataType pdt) [inline]

```

178
179     const int pdtBytes[] = {1, 2, 4};
180     return pdtBytes[(int)pdt];
181 }
```

11.33.3.4 void SetCUDADevices (int * devices, int numdev)

```

12
13 }
```

11.34 cputrack/random_distr.h File Reference

Functions

- template<typename T >
T [rand_uniform \(\)](#)
- template<typename T >
T [rand_normal \(\)](#)
- template<typename T >
int [rand_poisson \(T lambda\)](#)

11.34.1 Function Documentation

11.34.1.1 template<typename T > T rand_normal ()

```

12 {
13     T U0 = rand_uniform<T>()+(T)1e-9;
14     T U1 = rand_uniform<T>();
15
16     // Box-Muller transform
17     return sqrt( -2 * log(U0) ) * cos(2*(T)3.141593 * U1);
18 }
```

11.34.1.2 template<typename T > int rand_poisson (T lambda)

```

22 {
23     if (lambda > 10) {
24         T v = rand_normal<T>();
25         v = (T)0.5 + lambda + v*sqrt(lambda);
26         return (int)std::max((T)0,v);
27     }
28     else {
29         T L = exp(-lambda);
30         int k = 0;
31         T p = 1;
32         do {
33             k++;
34             T u = rand_uniform<T>();
35             p = p * u;
36         } while (p > L && k<200);
37
38         // copy back
39         return k-1;
40     }
41 }
```

11.34.1.3 template<typename T > T rand_uniform ()

```

5 {
6     return rand() / (T)RAND_MAX;
7 }
```

11.35 cputrack/ResultManager.cpp File Reference

```
#include "std_incl.h"
#include "ResultManager.h"
#include "utils.h"
```

Macros

- #define BINFILE_VERSION 3

11.35.1 Macro Definition Documentation

11.35.1.1 #define BINFILE_VERSION 3

11.36 cputrack/ResultManager.h File Reference

```
#include "QueuedTracker.h"
#include <list>
#include "threads.h"
```

Classes

- class [ResultFile](#)
- class [TextResultFile](#)
- class [BinaryResultFile](#)
- struct [ResultManagerConfig](#)
- class [ResultManager](#)
- struct [ResultManager::FrameCounters](#)
- struct [ResultManager::FrameResult](#)

11.37 cputrack/scalar_types.h File Reference

```
#include <complex>
#include <vector>
```

TypeDefs

- typedef float [scalar_t](#)
- typedef std::complex<[scalar_t](#)> [complex_t](#)

11.37.1 Typedef Documentation

11.37.1.1 [typedef std::complex<scalar_t> complex_t](#)

11.37.1.2 [typedef float scalar_t](#)

11.38 cputrack/std_incl.h File Reference

```
#include <crtdbg.h>
#include "memdbg.h"
#include <cstdint>
#include <string>
#include <deque>
#include <vector>
#include <algorithm>
#include <cstdio>
#include <stdexcept>
#include <cassert>
#include <cmath>
#include <cstdlib>
#include <cstddef>
#include <complex>
#include "random_distr.h"
#include "dllmacros.h"
```

Classes

- struct `vector2< T >`
- struct `vector3< T >`

Macros

- `#define _CRT_SECURE_NO_WARNINGS`
- `#define _CRT_SECURE_NO_WARNINGS`
- `#define STRCASECMP strcasecmp`
- `#define STRNCASECMP strncasecmp`
- `#define SNPRINTF sprintf`
- `#define VSNPRINTF vsprintf`
- `#define ALLOCA(size) alloca(size)`
- `#define ALLOCA_ARRAY(T, N) ((T*)ALLOCA(sizeof(T) * (N)))`

Typedefs

- `typedef vector2< float > vector2f`
- `typedef vector2< double > vector2d`
- `typedef vector3< float > vector3f`
- `typedef vector3< double > vector3d`
- `typedef unsigned int uint`
- `typedef unsigned short ushort`
- `typedef unsigned long ulong`
- `typedef unsigned char uchar`

Functions

- template<typename T>
`vector3< T > sqrt (const vector3< T > &a)`

11.38.1 Macro Definition Documentation

11.38.1.1 `#define _CRT_SECURE_NO_WARNINGS`

11.38.1.2 `#define _CRT_SECURE_NO_WARNINGS`

11.38.1.3 `#define ALLOCA(size) alloca(size)`

11.38.1.4 `#define ALLOCA_ARRAY(T, N) ((T*)ALLOCA(sizeof(T) * (N)))`

11.38.1.5 `#define SNPRINTF sprintf`

11.38.1.6 `#define STRCASECMP strcasecmp`

11.38.1.7 `#define STRNCASECMP strncasecmp`

11.38.1.8 #define VSNPRINTF vsnprintf

11.38.2 Typedef Documentation

11.38.2.1 `typedef unsigned char uchar`

11.38.2.2 `typedef unsigned int uint`

11.38.2.3 `typedef unsigned long ulong`

11.38.2.4 `typedef unsigned short ushort`

11.38.2.5 `typedef vector2<double> vector2d`

11.38.2.6 `typedef vector2<float> vector2f`

11.38.2.7 `typedef vector3<double> vector3d`

11.38.2.8 `typedef vector3<float> vector3f`

11.38.3 Function Documentation

11.38.3.1 `template<typename T > vector3<T> sqrt(const vector3<T> & a) [inline]`

```
112 { return vector3<T>(sqrt(a.x),sqrt(a.y),sqrt(a.z)); }
```

11.39 cputrack/threads.h File Reference

```
#include <list>
#include <Windows.h>
```

Classes

- struct `Threads`
- struct `Threads::Handle`
- struct `Threads::Mutex`
- class `Atomic< T >`
- class `ThreadPool< TWorkItem, TFunctor >`

TypeDefs

- `typedef Threads::Handle ThreadHandle`

Functions

- template<typename TF >
void [parallel_for](#) (int count, TF f)

11.39.1 Typedef Documentation

11.39.1.1 [typedef Threads::Handle ThreadHandle](#)

11.39.2 Function Documentation

11.39.2.1 template<typename TF > void [parallel_for](#) (int count, TF f)

```

251
252
253     if (count == 1)
254         f(0);
255     else {
256         ThreadPool<int, TF> threadPool(f, std::min (count,
257             Threads::GetCPUCount()) );
258         for (int i=0;i<count;i++) threadPool.AddWork(i);
259         threadPool.WaitUntilDone();
260     }
260 }
```

11.40 cputrack/utils.cpp File Reference

```
#include "std_incl.h"
#include <cstdarg>
#include <functional>
#include "utils.h"
#include <Windows.h>
#include <string>
#include <complex>
#include "random_distr.h"
#include "LsqQuadraticFit.h"
#include "QueuedTracker.h"
#include "threads.h"
#include "CubicBSpline.h"
#include "time.h"
#include <tchar.h>
```

Functions

- std::string [GetCurrentOutputPath](#) (bool ext)
- void [GetFormattedTimeString](#) (char *output)
- void [dbgsetlogfile](#) (const char *path)
- std::string [GetLocalModuleFilename](#) ()
- void [WriteToLog](#) (const char *str)
- std::string [GetDirectoryFromPath](#) (std::string fullpath)
- std::string [GetLocalModulePath](#) ()
- std::string [file_ext](#) (const char *f)
- std::string [SPrintf](#) (const char *fmt,...)

- void `dbgout` (const std::string &s)
- void `dbgprintf` (const char *fmt,...)
- void `GenerateTestImage` (ImageData &img, float xp, float yp, float size, float SNratio)
- void `ComputeCRP` (float *dst, int radialSteps, int angularSteps, float minradius, float maxradius, `vector2f` center, ImageData *img, float paddingValue, float *crpmap)
- float `ComputeBgCorrectedCOM1D` (float *data, int len, float cf)
- void `NormalizeRadialProfile` (`scalar_t` *prof, int rsteps)
- void `NormalizeZLUT` (float *zlut, int numBeads, int planes, int radialsteps)
- void `ComputeRadialProfile` (float *dst, int radialSteps, int angularSteps, float minradius, float maxradius, `vector2f` center, ImageData *img, float mean, bool `normalize`)
- float `sq` (float x)
- void `GenerateImageFromLUT` (ImageData *image, ImageData *zlut, float minradius, float maxradius, `vector3f` pos, bool splineInterp, int oversampleSubdiv)
- void `GenerateGaussianSpotImage` (ImageData *img, `vector2f` pos, float sigma, float I0, float Ibg)
- void `ApplyPoissonNoise` (ImageData &img, float poissonMax, float maxval)
- void `ApplyGaussianNoise` (ImageData &img, float sigma)
- std::vector< std::vector< float > > `ReadCSV` (const char *filename, char sep)
- std::vector< `vector3f` > `ReadVector3CSV` (const char *file, char sep)
- void `WriteTrace` (std::string filename, `vector3f` *results, int nResults)
- void `WriteVectorAsCSVRow` (const char *file, std::vector< float > d, bool append)
- void `WriteArrayAsCSVRow` (const char *file, float *d, int len, bool append)
- void `WriteImageAsCSV` (const char *file, float *d, int w, int h, const char *labels[])
- void `WriteComplexImageAsCSV` (const char *file, std::complex< float > *d, int w, int h, const char *labels[])
- std::vector< uchar > `ReadToByteBuffer` (const char *filename)
- `ImageData ReadJPEGFile` (const char *fn)
- void `CopyImageToFloat` (uchar *data, int width, int height, int pitch, `QTRK_PixelDataType` pdt, float *dst)
- double `GetPreciseTime` ()
- int `NearestPowerOf2` (int v)
- int `NearestPowerOf3` (int v)
- std::vector< float > `ComputeRadialBinWindow` (int rsteps)
- `ImageData ReadLUTFile` (const char *lutfile)

Variables

- static std::string `logFilename`

11.40.1 Function Documentation

11.40.1.1 void `ApplyGaussianNoise` (`ImageData & img`, `float sigma`)

```

459 {
460     for (int k=0;k

```

11.40.1.2 void ApplyPoissonNoise (*ImageData & img*, float poissonMax, float maxval)

```

437 {
438     /*auto f = [&] (int y) {
439         for (int x=0;x

```

11.40.1.3 float ComputeBgCorrectedCOM1D (float * *data*, int *len*, float *cf*)

```

234 {
235     float sum=0, sum2=0;
236     float moment=0;
237
238     for (int x=0;x<len;x++) {
239         float v = data[x];
240         sum += v;
241         sum2 += v*v;
242     }
243
244     float invN = 1.0f/len;
245     float mean = sum * invN;
246     float stdev = sqrtf(sum2 * invN - mean * mean);
247     sum = 0.0f;
248
249     for(int x=0;x<len;x++)
250     {
251         float v = data[x];
252         v = std::max(0.0f, fabs(v-mean)-cf*stdev);
253         sum += v;
254         moment += x*v;
255     }
256     return moment / (float)sum;
257 }
```

11.40.1.4 void ComputeCRP (float * *dst*, int *radialSteps*, int *angularSteps*, float *minradius*, float *maxradius*, vector2f *center*, *ImageData* * *img*, float *paddingValue*, float * *crpmap*)

```

185 {
186     vector2f* radialDirs = (vector2f*)ALLOCA(sizeof(
187     vector2f)*angularSteps);
188     for (int j=0;j<angularSteps;j++) {
189         float ang = 2*3.141593f*j/(float)angularSteps;
190         radialDirs[j] = vector2f(cosf(ang), sinf(ang) );
191     }
192
193     for (int i=0;i<radialSteps;i++)
194         dst[i]=0.0f;
195
196     float* map = crpmap ? crpmap : (float*)ALLOCA(sizeof(float)*radialSteps*angularSteps);
197     float* com = (float*)ALLOCA(sizeof(float)*angularSteps);
198
199     float rstep = (maxradius-minradius) / radialSteps;
200     float comsum = 0.0f;
201     for (int a=0;a<angularSteps;a++) {
202         float r = minradius;
203         float sum = 0.0f, moment=0.0f;
204         for (int i=0;i<radialSteps; i++) {
205             float x = center.x + radialDirs[a].x * r;
206             float y = center.y + radialDirs[a].y * r;
```

```

206         float v = img->interpolate(x,y);
207         r += rstep;
208         map[a*radialSteps+i] = v;
209         sum += v;
210         moment += i*v;
211     }
212     com[a] = moment/sum;
213     comsum += com[a];
214 }
215 float avgcom = comsum/angularSteps;
216 float totalrmssum2 = 0.0f;
217 for (int i=0;i<radialSteps; i++) {
218     double sum = 0.0f;
219     for (int a=0;a<angularSteps;a++) {
220         float shift = com[a]-avgcom;
221         sum += map[a*radialSteps+i];
222     }
223     dst[i] = sum/angularSteps-paddingValue;
224     totalrmssum2 += dst[i]*dst[i];
225 }
226 double invTotalrms = 1.0f/sqrt(totalrmssum2/radialSteps);
227 for (int i=0;i<radialSteps;i++) {
228     dst[i] *= invTotalrms;
229 }
230 }
```

11.40.1.5 std::vector<float> ComputeRadialBinWindow (int *rsteps*)

```

714 {
715     std::vector<float> wnd(rsteps);
716     for (int i=0;i<rsteps;i++) {
717         float x = i/(float)rsteps;
718         float t2 = 0.05f;
719         float t1 = 0.01f;
720         float fall = 1.0f-expf(-sq(1-x)/t2);
721         float rise = 1.0f-expf(-sq(x)/t1);
722         wnd[i] = sqrtf(fall*rise*x*x);
723     }
724     return wnd;
725 }
```

11.40.1.6 void ComputeRadialProfile (float * *dst*, int *radialSteps*, int *angularSteps*, float *minradius*, float *maxradius*, vector2f *center*, ImageData * *img*, float *mean*, bool *normalize*)

```

302 {
303     vector2f* radialDirs = (vector2f*)ALLOCA(sizeof(
vector2f)*angularSteps);
304     for (int j=0;j<angularSteps; j++) {
305         float ang = 2*3.141593f*j/(float)angularSteps;
306         radialDirs[j] = vector2f(cosf(ang), sinf(ang));
307     }
308     for (int i=0;i<radialSteps;i++)
309         dst[i]=0.0f;
310     // center.x += 0.5f;
311     // center.y += 0.5f;
312     bool trace=false;
313     float rstep = (maxradius-minradius) / radialSteps;
314     int totalsmp = 0;
315     //FILE* f = fopen("D:\\TestImages\\test.csv","w");
316     for (int i=0;i<radialSteps; i++) {
317         double sum = 0.0f;
318         int nsamples = 0;
319         float r = minradius+rstep*i;
320         for (int a=0;a<angularSteps;a++) {
321             float x = center.x + radialDirs[a].x * r;
322             float y = center.y + radialDirs[a].y * r;
323             bool outside;
324             float v = img->interpolate(x,y, &outside);
325             if (!outside) {
326                 sum += v;
327             }
328         }
329     }
330 }
```

```

333             nsamples++;
334         }
335
336         //fprintf(f,"%d\t%d\t%f\n",i,a,v);
337     }
338
339     if (trace) {
340         dbgprintf("%f,[%d]; ", sum, nsamples);
341     }
342
343     dst[i] = nsamples > MIN_RADPROFILE_SMP_COUNT ? sum/nsamples : mean;
344 }
345 if(trace)
346     dbgprintf("\n");
347
348 //fclose(f);
349
350 if (normalize)
351     NormalizeRadialProfile(dst, radialSteps);
352 }
```

11.40.1.7 void CopyImageToFloat (uchar * data, int width, int height, int pitch, QTRK_PixelDataType pdt, float * dst)

```

650 {
651     if (pdt == QTrkU8) {
652         for (int y=0;y<height;y++) {
653             for (int x=0;x<width;x++)
654                 dst[x] = data[x];
655             data += pitch;
656             dst += width;
657         }
658     } else if(pdt == QTrkU16) {
659         for (int y=0;y<height;y++) {
660             ushort* u = (ushort*)data;
661             for (int x=0;x<width;x++)
662                 dst[x] = u[x];
663             data += pitch;
664             dst += width;
665         }
666     } else {
667         for (int y=0;y<height;y++) {
668             float* fsrc = (float*)data;
669             for( int x=0;x<width;x++)
670                 dst[x] = fsrc[x];
671             data += pitch;
672             dst += width;
673         }
674     }
675 }
```

11.40.1.8 void dbgout (const std::string & s)

```

143
144     OutputDebugString(s.c_str());
145     printf(s.c_str());
146     WriteToLog(s.c_str());
147 }
```

11.40.1.9 void dbgprintf (const char * fmt, ...)

```

149
150     va_list ap;
151     va_start(ap, fmt);
152
153     char buf[512];
154     VSNPRINTF(buf, sizeof(buf), fmt, ap);
155     OutputDebugString(buf);
156     fputs(buf,stdout);
157     WriteToLog(buf);
158
159     va_end(ap);
160 }
```

11.40.1.10 void dbgsetlogfile (const char * path)

```
50 {
51     logfilename = path;
52 }
```

11.40.1.11 std::string file_ext (const char * f)

```
121 {
122     int l=strlen(f)-1;
123     while (l > 0) {
124         if (f[l] == '.')
125             return &f[l+1];
126         l--;
127     }
128     return "";
129 }
```

11.40.1.12 void GenerateGaussianSpotImage (ImageData * img, vector2f pos, float sigma, float I0, float Ibg)

```
426 {
427     float edenom = 1/sqrt(2*sigma*sigma);
428     for (int y=0;y

```

11.40.1.13 void GenerateImageFromLUT (ImageData * image, ImageData * zlut, float minradius, float maxradius, vector3f pos, bool splineInterp, int oversampleSubdiv)

```
358 {
359 //    lut.w = radialcov * (image->w/2 * roicov ) - minradius );
360 //    lut.w = radialcov * ( maxradius - minradius );
361
362     float radialcov = zlut->w / (maxradius-minradius);
363     float* zinterp = (float*)ALLOC(zlut->w * sizeof(float));
364
365     if (splineInterp) {
366         int iz = std::max(1, std::min(zlut->h-3, (int)pos.z));
367         float weights[4];
368         float fz = pos.z-iz;
369         ComputeBSplineWeights(weights, fz);
370         // Interpolate ZLUT using B-spline weights
371         for (int r=0;r

```

```

392     }
393
394     int oversampleWidth=oversampleSubdiv,oversampleHeight=oversampleSubdiv;
395     float oxstep = 1.0f / oversampleWidth;
396     float oystep = 1.0f / oversampleHeight;
397
398     for (int y=0;y<image->h;y++)
399         for (int x=0;x<image->w;x++)
400     {
401         float s = 0.0f;
402
403         for (int ox=0;ox<oversampleWidth;ox++)
404             for (int oy=0;oy<oversampleHeight;oy++) {
405
406                 float X = x+(ox+0.5f)*oxstep - pos.x - 0.5f;
407                 float Y = y+(oy+0.5f)*oystep - pos.y - 0.5f;
408
409                 float pixr = sqrtf(X*X+Y*Y);
410                 float r = (pixr - minradius) * radialcov;
411
412                 if (r > zlut->w-2)
413                     r = zlut->w-2;
414                 if (r < 0) r = 0;
415
416                 int i=(int)r;
417                 s += Lerp(zinterp[i], zinterp[i+1], r-i);
418             }
419
420             image->at(x,y) = s/(oversampleWidth*oversampleHeight);
421         }
422     }

```

11.40.1.14 void GenerateTestImage (**ImageData & img**, float *xp*, float *yp*, float *size*, float *SNratio*)

```

163 {
164     float S = 1.0f/sqrt(size);
165     for (int y=0;y<img.h;y++) {
166         for (int x=0;x<img.w;x++) {
167             float X = x - xp;// + 0.5;
168             float Y = y - yp;// + 0.5;
169             float r = sqrtf(X*X+Y*Y)+1;
170             float v = sinf(r/(5*S)) * expf(-r*r*S*0.001f);
171             img.at(x,y)=v;
172         }
173     }
174
175     if (SNratio>0) {
176         ApplyGaussianNoise(img, 1.0f/SNratio);
177     }
178     img.normalize();
179 }

```

11.40.1.15 std::string GetCurrentOutputPath (bool *ext*)

```

20 {
21     std::string base = "D:\\TestImages\\TestOutput\\";
22     std::string search = base + "*";
23     LPCSTR w_folder = _T(search.c_str());
24
25     WIN32_FIND_DATA FindFileData;
26     HANDLE hFind;
27
28     hFind = FindFirstFile(w_folder,&FindFileData);
29     std::string dirName;
30     while(FindNextFile(hFind,&FindFileData))
31         dirName = FindFileData.cFileName;
32     if(ext)
33         return SPrintf("%s%s\\%s",base.c_str(),dirName.c_str(),"ZLUTDiag\\");
34     else
35         return SPrintf("%s%s",base.c_str(),dirName.c_str());
36 }

```

11.40.1.16 std::string GetDirectoryFromPath (std::string *fullpath*)

```

105 {
106     for (int i=fullpath.size()-1;i>=0;i--) {
107         if (fullpath[i] == '/') || fullpath[i] == '\\') {
108             return fullpath.substr(0, i);
109         }
110     }
111     return "";
112 }
```

11.40.1.17 void GetFormattedTimeString (char * *output*)

```

39 {
40     time_t rawtime;
41     struct tm * timeinfo;
42     time(&rawtime);
43     timeinfo = localtime(&rawtime);
44     sprintf(output, "%02d%02d%02d-%02d%02d%02d", timeinfo->tm_year-100, timeinfo->tm_mon+1, timeinfo->tm_mday,
45         timeinfo->tm_hour, timeinfo->tm_min, timeinfo->tm_sec);
46 }
```

11.40.1.18 std::string GetLocalModuleFilename ()

```

55 {
56 #ifdef WIN32
57     char path[256];
58     HMODULE hm = NULL;
59     GetModuleHandleExA(GET_MODULE_HANDLE_EX_FLAG_FROM_ADDRESS |
60     GET_MODULE_HANDLE_EX_FLAG_UNCHANGED_REFCOUNT,
61     (LPCSTR) &GetLocalModuleFilename, &hm);
62     GetModuleFileNameA(hm, path, sizeof(path));
63     return path;
64 #else
65 #error GetLocalModuleName() not implemented for this platform
66 #endif
67 }
68 }
```

11.40.1.19 std::string GetLocalModulePath ()

```

115 {
116     std::string dllpath = GetLocalModuleFilename();
117     return GetDirectoryFromPath(dllpath);
118 }
```

11.40.1.20 double GetPreciseTime ()

```

680 {
681     uint64_t freq, time;
682     QueryPerformanceCounter((LARGE_INTEGER*)&time);
683     QueryPerformanceFrequency((LARGE_INTEGER*)&freq);
684     return (double)time / (double)freq;
685 }
686 }
```

11.40.1.21 int NearestPowerOf2(int v)

```

693 {
694     int r=1;
695     while (r < v)
696         r *= 2;
697     if ( fabsf(r-v) < fabsf(r/2-v) )
698         return r;
699     return r/2;
700 }

```

11.40.1.22 int NearestPowerOf3(int v)

```

703 {
704     int r=1;
705     while (r < v)
706         r *= 3;
707     if ( fabsf(r-v) < fabsf(r/3-v) )
708         return r;
709     return r/3;
710 }

```

11.40.1.23 void NormalizeRadialProfile(scalar_t * prof, int rsteps)

```

260 {
261     double sum=0.0f;
262     for (int i=0;i<rsteps;i++)
263         sum += prof[i];
264
265     float mean =sum/rsteps;
266     double rmssum2 = 0.0;
267
268     for (int i=0;i<rsteps;i++) {
269         prof[i] -= mean;
270         rmssum2 += prof[i]*prof[i];
271     }
272     double invTotalrms = 1.0f/sqrt(rmssum2/rsteps);
273     for (int i=0;i<rsteps;i++)
274         prof[i] *= invTotalrms;
275
276 /*
277     scalar_t minValue = prof[0];
278     for (int i=0;i<rsteps;i++)
279         if(prof[i]<minValue) minValue=prof[i];
280
281     float rms=0;
282     for (int i=0;i<rsteps;i++) {
283         prof[i]-=minValue;
284         rms += prof[i]*prof[i];
285     }
286     rms=1.0f/sqrt(rms);
287     for (int i=0;i<rsteps;i++) {
288         prof[i]*=rms;
289     }*/
290 }

```

11.40.1.24 void NormalizeZLUT(float * zlut, int numBeads, int planes, int radialsteps)

```

294 {
295     for(int i=0;i<numBeads;i++)
296         for (int j=0;j<planes;j++)
297             NormalizeRadialProfile(&zlut[radialsteps*planes*i + radialsteps*j],
298                                     radialsteps);
298 }

```

11.40.1.25 std::vector< std::vector<float> > ReadCSV (const char * *filename*, char *sep*)

```

468 {
469     std::list< std::vector <float> > data;
470
471     FILE *f=fopen(filename,"r");
472     if (f) {
473         std::string buf;
474         std::vector<float> vals;
475         while (!feof(f)) {
476             char c=fgetc(f);
477             if (c == sep || c=='\n') {
478                 vals.push_back( atof(buf.c_str()) );
479                 buf.clear();
480             }
481             if (c == '\n') {
482                 data.push_back( vals );
483                 vals.clear();
484             }
485             if (c != sep && c!='\n')
486                 buf+=c;
487         }
488         fclose(f);
489     }
490
491     std::vector<std::vector<float> > r;
492     r.reserve(data.size());
493     r.insert(r.begin(), data.begin(), data.end());
494     return r;
495 }
496 }
```

11.40.1.26 ImageData ReadJPEGFile (const char * *fn*)

```

634 {
635     int w, h;
636     uchar* imgdata;
637     std::vector<uchar> jpgdata = ReadToByteBuffer(fn);
638     ReadJPEGFile(&jpgdata[0], jpgdata.size(), &imgdata, &w,&h);
639
640     float* fbuf = new float[w*h];
641     for (int x=0;x<w*h;x++)
642         fbuf[x] = imgdata[x]/255.0f;
643     delete[] imgdata;
644
645     return ImageData(fbuf,w,h);
646 }
```

11.40.1.27 ImageData ReadLUTFile (const char * *lutfile*)

```

729 {
730     PathSeparator sep(lutfile);
731     if(sep.extension == "jpg") {
732         return ReadJPEGFile(lutfile);
733     }
734     else {
735         std::string fn = lutfile;
736         fn = std::string(fn.begin(), fn.begin()+fn.find('#'));
737         std::string num( ++( sep.extension.begin() + sep.extension.find('#') ), sep.extension.end());
738         int lutIndex = atoi(num.c_str());
739
740         int nbeads, nplanes, nsteps;
741         FILE *f = fopen(fn.c_str(), "rb");
742
743         if (!f)
744             throw std::runtime_error("Can't open " + fn);
745
746         fread(&nbeads, 4, 1, f);
747         fread(&nplanes, 4, 1, f);
748         fread(&nsteps, 4, 1, f);
749
750
751         fseek(f, 12 + 4* (nsteps*nplanes * lutIndex), SEEK_SET);
752         ImageData lut = ImageData::alloc(nsteps,nplanes);
753         fread(lut.data, 4, nsteps*nplanes,f);
754         fclose(f);
755         lut.normalize();
756         return lut;
757     }
758 }
```

11.40.1.28 std::vector<uchar> ReadToByteBuffer(const char * *filename*)

```

615 {
616     FILE *f = fopen(filename, "rb");
617
618     if (!f)
619         throw std::runtime_error(SPrintf("%s was not found", filename));
620
621     fseek(f, 0, SEEK_END);
622     int len = ftell(f);
623     fseek(f, 0, SEEK_SET);
624
625     std::vector<uchar> buf(len);
626     fread(&buf[0], 1, len, f);
627
628     fclose(f);
629     return buf;
630 }
```

11.40.1.29 std::vector<vector3f> ReadVector3CSV(const char * *file*, char *sep*)

```

500 {
501     auto data=ReadCSV(file ,sep);
502
503     std::vector<vector3f> r(data.size());
504
505     for (int i=0;i<data.size();i++){
506         r[i]=vector3f(data[i][0],data[i][1],data[i][2]);
507     }
508     return r;
509 }
```

11.40.1.30 std::string SPrintf(const char * *fmt*, ...)

```

132 {
133     va_list ap;
134     va_start(ap, fmt);
135
136     char buf[512];
137     VSNPRINTF(buf, sizeof(buf), fmt, ap);
138
139     va_end(ap);
140     return buf;
141 }
```

11.40.1.31 float sq(float *x*) [inline]

```
355 { return x*x; }
```

11.40.1.32 void WriteArrayAsCSVRow(const char * *file*, float * *d*, int *len*, bool *append*)

```

543 {
544     FILE *f = fopen(file, append?"a":"w");
545     if(f) {
546         for (int i=0;i<len;i++)
547             fprintf(f, "% .7f\t", d[i]);
548
549         fprintf(f, "\n");
550         fclose(f);
551     }
552     else
553         dbgprintf("WriteArrayAsCSVRow: Unable to open file %s\n", file);
554 }
```

11.40.1.33 void WriteComplexImageAsCSV (const char * file, std::complex< float > * d, int w, int h, const char * labels[])

```

586 {
587     FILE* f = fopen(file, "w");
588
589     if (!f) {
590         dbgprintf("WriteComplexImageAsCSV: Unable to open file %s\n", file);
591         return;
592     }
593
594     if (labels) {
595         for (int i=0;i<w;i++) {
596             fprintf(f, "%s;\t", labels[i]);
597         }
598         fputs("\n", f);
599     }
600
601     for (int y=0;y<h;y++) {
602         for (int x=0;x<w;x++) {
603             {
604                 float i=d[y*w+x].imag();
605                 fprintf(f, "%f%+fi", d[y*w+x].real(), i);
606                 if(x<w-1) fputs("\t", f);
607             }
608             fprintf(f, "\n");
609         }
610     }
611     fclose(f);
612 }
```

11.40.1.34 void WriteImageAsCSV (const char * file, float * d, int w, int h, const char * labels[])

```

557 {
558     FILE* f = fopen(file, "w");
559
560     if (f) {
561
562         if (labels) {
563             for (int i=0;i<w;i++) {
564                 fprintf(f, "%s;\t", labels[i]);
565             }
566             fputs("\n", f);
567         }
568
569         for (int y=0;y<h;y++) {
570             for (int x=0;x<w;x++) {
571                 {
572                     fprintf(f, "%10f", d[y*w+x]);
573                     if(x<w-1) fputs("\t", f);
574                 }
575                 fprintf(f, "\n");
576             }
577         }
578     }
579     fclose(f);
580 } else
581     dbgprintf("WriteImageAsCSV: Unable to open file %s\n", file);
582 }
```

11.40.1.35 void WriteToLog (const char * str)

```

89 {
90     if (logFilename.empty()) {
91         auto ps = PathSeparator(GetLocalModuleFilename());
92         logFilename = ps.directory + ps.filename + "-log.txt";
93     }
94
95     if (str) {
96         FILE* f = fopen(logFilename.c_str(), "a");
97         if (f) {
98             fputs(str, f);
99             fclose(f);
100        }
101    }
102 }
```

11.40.1.36 void WriteTrace (std::string *filename*, vector3f * *results*, int *nResults*)

```

513 {
514     FILE *f = fopen(filename.c_str(), "w");
515
516     if (!f) {
517         throw std::runtime_error(SPrintf("Can't open %s", filename.c_str()));
518     }
519
520     for (int i=0;i<nResults;i++)
521     {
522         fprintf(f, "% .7f\t% .7f\t% .7f\n", results[i].x, results[i].y, results[i].z);
523     }
524
525     fclose(f);
526 }
```

11.40.1.37 void WriteVectorAsCSVRow (const char * *file*, std::vector< float > *d*, bool *append*)

```

529 {
530     FILE *f = fopen(file, append?"a":"w");
531     if(f) {
532         for (int i=0;i<d.size();i++)
533             fprintf(f, "%1.7f\t", d[i]);
534
535         fprintf(f, "\n");
536         fclose(f);
537     }
538     else
539         dbgprintf("WriteArrayAsCSVRow: Unable to open file %s\n", file);
540 }
```

11.40.2 Variable Documentation

11.40.2.1 std::string logFilename [static]

11.41 cputrack/utils.h File Reference

```
#include "std_incl.h"
#include "scalar_types.h"
```

Classes

- struct [TImageData< T >](#)
- class [CImageData](#)
- struct [PathSeperator](#)
- class [Matrix3X3](#)

Typedefs

- typedef [TImageData< float >](#) [ImageData](#)
- typedef [TImageData< double >](#) [ImageDatad](#)

Functions

- template<typename T >
 bool **isNaN** (const T &v)
- void **GetFormattedTimeString** (char *output)
- std::string **GetCurrentOutputPath** (bool ext=true)
- void **dbgout** (const std::string &s)
- std::string **SPrintf** (const char *fmt,...)
- void **dbgprintf** (const char *fmt,...)
- void **dbgsetlogfile** (const char *path)
- template<typename T >
 void **DeleteAllElems** (T &c)
- template<typename TPixel >
 void **normalize** (TPixel *d, uint w, uint h)
- template<typename T >
 T **Lerp** (T a, T b, float x)
- template<typename T >
 T **Interpolate** (T *image, int width, int height, float x, float y, bool *outside=0)
- template<typename T >
 T **Interpolate1D** (T *d, int len, float x)
- template<typename T >
 T **Interpolate1D** (const std::vector< T > &d, float x)
- template<typename T >
 T **StdDeviation** (T *start, T *end)
- std::vector< float > **ComputeRadialBinWindow** (int rsteps)
- float **ComputeBgCorrectedCOM1D** (float *data, int len, float cf=2.0f)
- void **ComputeCRP** (float *dst, int radialSteps, int angularSteps, float minradius, float maxradius, **vector2f** center, **ImageData** *src, float mean, float *crpmap=0)
- void **ComputeRadialProfile** (float *dst, int radialSteps, int angularSteps, float minradius, float maxradius, **vector2f** center, **ImageData** *src, float mean, bool **normalize**)
- void **NormalizeRadialProfile** (float *prof, int rsteps)
- void **NormalizeZLUT** (float *zlut, int numLUTs, int planes, int radialsteps)
- void **GenerateImageFromLUT** (**ImageData** *image, **ImageData** *zlut, float minradius, float maxradius, **vector3f** pos, bool useSplineInterp=true, int ovs=4)
- void **ApplyPoissonNoise** (**ImageData** &img, float poissonMax, float maxValue=255)
- void **ApplyGaussianNoise** (**ImageData** &img, float sigma)
- void **WriteComplexImageAsCSV** (const char *file, std::complex< float > *d, int w, int h, const char *labels[] = 0)
- void **WriteArrayAsCSVRow** (const char *file, float *d, int len, bool append)
- void **WriteVectorAsCSVRow** (const char *file, std::vector< float > d, bool append)
- void **WriteImageAsCSV** (const char *file, float *d, int w, int h, const char *labels[] = 0)
- std::vector< std::vector< float > > **ReadCSV** (const char *filename, char sep='t')
- std::vector< **vector3f** > **ReadVector3CSV** (const char *file, char sep='t')
- void **WriteTrace** (std::string file, **vector3f** *results, int nResults)
- void **GenerateTestImage** (**ImageData** &img, float xp, float yp, float size, float MaxPhotons)
- std::string **GetLocalModuleFilename** ()
- std::string **GetLocalModulePath** ()
- std::string **GetDirectoryFromPath** (std::string fullpath)
- std::string **file_ext** (const char *f)
- **ImageData** **ReadJPEGFile** (const char *fn)
- **ImageData** **ReadLUTFile** (const char *lutfile)
- int **ReadJPEGFile** (uchar *srcbuf, int srclen, uchar **data, int *width, int *height)
- void **WriteJPEGFile** (uchar *data, int w, int h, const char *filename, int quality)
- void **FloatToJPEGFile** (const char *name, const float *d, int w, int h)
- void **WriteJPEGFile** (const char *name, const **ImageData** &img)

- int `NearestPowerOf2` (int v)
- int `NearestPowerOf3` (int v)
- void `GenerateGaussianSpotImage` (`ImageData` *img, `vector2f` pos, float sigma, float l0, float lbg)
- std::vector< uchar > `ReadToByteBuffer` (const char *filename)
- double `GetPreciseTime` ()
- template<typename T >
void `floatToNormalizedInt` (T *dst, const float *src, uint w, uint h, T maxValue)
- template<typename T >
T * `floatToNormalizedInt` (const float *src, uint w, uint h, T maxValue)
- template<typename T >
T `ComputeStdDev` (T *data, int len)
- template<typename T >
T `qselect` (T *data, int start, int end, int k)
- template<typename T >
T `erf` (T x)

11.41.1 Typedef Documentation

11.41.1.1 `typedef TImageData<float> ImageData`

11.41.1.2 `typedef TImageData<double> ImageDataad`

11.41.2 Function Documentation

11.41.2.1 `void ApplyGaussianNoise (ImageData & img, float sigma)`

```
459 {
460     for (int k=0;k.numPixels();k++) {
461         float v = img.data[k] + sigma * rand_normal<float>();
462         if (v<0.0f) v= 0.0f;
463         img.data[k]=v;
464     }
465 }
```

11.41.2.2 `void ApplyPoissonNoise (ImageData & img, float poissonMax, float maxValue = 255)`

```
437 {
438     /*auto f = [&] (int y) {
439         for (int x=0;x.w;x++) {
440             img.at(x,y) = rand_poisson<float>(factor.at(x,y));
441         }
442     };
443     ThreadPool<int, std::function<void (int index)> > pool(f);
444     for (int y=0;y.h;y++) {
445         pool.AddWork(y);
446     }
447     pool.WaitUntilDone();*/
450     float ratio = maxval / poissonMax;
452     for (int x=0;x.numPixels();x++) {
453         img[x] = (int)(rand_poisson<float>(poissonMax[x]) * ratio );
455     }
456 }
```

11.41.2.3 float ComputeBgCorrectedCOM1D (float * *data*, int *len*, float *cf*=2.0f)

```

234 {
235     float sum=0, sum2=0;
236     float moment=0;
237
238     for (int x=0;x<len;x++) {
239         float v = data[x];
240         sum += v;
241         sum2 += v*v;
242     }
243
244     float invN = 1.0f/len;
245     float mean = sum * invN;
246     float stdev = sqrtf(sum2 * invN - mean * mean);
247     sum = 0.0f;
248
249     for(int x=0;x<len;x++)
250     {
251         float v = data[x];
252         v = std::max(0.0f, fabs(v-mean)-cf*stdev);
253         sum += v;
254         moment += x*v;
255     }
256     return moment / (float)sum;
257 }
```

11.41.2.4 void ComputeCRP (float * *dst*, int *radialSteps*, int *angularSteps*, float *minradius*, float *maxradius*, vector2f *center*, ImageData * *src*, float *mean*, float * *crpmap* = 0)

```

185 {
186     vector2f* radialDirs = (vector2f*)ALLOCA(sizeof(
187     vector2f)*angularSteps);
188     for (int j=0;j<angularSteps;j++) {
189         float ang = 2*3.141593f*j/(float)angularSteps;
190         radialDirs[j] = vector2f(cosf(ang), sinf(ang) );
191     }
192     for (int i=0;i<radialSteps;i++)
193         dst[i]=0.0f;
194
195     float* map = crpmap ? crpmap : (float*)ALLOCA(sizeof(float)*radialSteps*angularSteps);
196     float* com = (float*)ALLOCA(sizeof(float)*angularSteps);
197
198     float rstep = (maxradius-minradius) / radialSteps;
199     float comsum = 0.0f;
200     for (int a=0;a<angularSteps;a++) {
201         float r = minradius;
202         float sum = 0.0f, moment=0.0f;
203         for (int i=0;i<radialSteps; i++) {
204             float x = center.x + radialDirs[a].x * r;
205             float y = center.y + radialDirs[a].y * r;
206             float v = img->interpolate(x,y);
207             r += rstep;
208             map[a*radialSteps+i] = v;
209             sum += v;
210             moment += i*v;
211         }
212         com[a] = moment/sum;
213         comsum += com[a];
214     }
215     float avgcom = comsum/angularSteps;
216     float totalrmssum2 = 0.0f;
217     for (int i=0;i<radialSteps; i++) {
218         double sum = 0.0f;
219         for (int a=0;a<angularSteps;a++) {
220             float shift = com[a]-avgcom;
221             sum += map[a*radialSteps+i];
222         }
223         dst[i] = sum/angularSteps-paddingValue;
224         totalrmssum2 += dst[i]*dst[i];
225     }
226     double invTotalrms = 1.0f/sqrt(totalrmssum2/radialSteps);
227     for (int i=0;i<radialSteps;i++) {
228         dst[i] *= invTotalrms;
229     }
230 }
```

11.41.2.5 `std::vector<float> ComputeRadialBinWindow (int rsteps)`

```

714 {
715     std::vector<float> wnd(rsteps);
716     for (int i=0;i<rsteps;i++) {
717         float x = i/(float)rsteps;
718         float t2 = 0.05f;
719         float t1 = 0.01f;
720         float fall = 1.0f-expf(-sq(1-x)/t2);
721         float rise = 1.0f-expf(-sq(x)/t1);
722         wnd[i] = sqrtf(fall*rise*x*x);
723     }
724     return wnd;
725 }
```

11.41.2.6 `void ComputeRadialProfile (float * dst, int radialSteps, int angularSteps, float minradius, float maxradius, vector2f center, ImageData * src, float mean, bool normalize)`

```

302 {
303     vector2f* radialDirs = (vector2f*)ALLOCA(sizeof(
304     vector2f)*angularSteps);
305     for (int j=0;j<angularSteps;j++) {
306         float ang = 2*3.141593f*j/(float)angularSteps;
307         radialDirs[j] = vector2f(cosf(ang), sinf(ang));
308     }
309     for (int i=0;i<radialSteps;i++)
310         dst[i]=0.0f;
311
312 //    center.x += 0.5f;
313 //    center.y += 0.5f;
314
315     bool trace=false;
316     float rstep = (maxradius-minradius) / radialSteps;
317     int totalsmp = 0;
318
319 //FILE* f = fopen("D:\\TestImages\\test.csv","w");
320
321     for (int i=0;i<radialSteps; i++) {
322         double sum = 0.0f;
323
324         int nsamples = 0;
325         float r = minradius+rstep*i;
326         for (int a=0;a<angularSteps;a++) {
327             float x = center.x + radialDirs[a].x * r;
328             float y = center.y + radialDirs[a].y * r;
329             bool outside;
330             float v = img->interpolate(x,y, &outside);
331             if (!outside) {
332                 sum += v;
333                 nsamples++;
334             }
335
336             //fprintf(f,"%d\t%d\t%f\n",i,a,v);
337         }
338
339         if (trace) {
340             dbgprintf("%f,[%d]; ", sum, nsamples);
341         }
342
343         dst[i] = nsamples > MIN_RADPROFILE_SMP_COUNT ? sum/nsamples : mean;
344     }
345     if(trace)
346         dbgprintf("\n");
347
348 //fclose(f);
349
350     if (normalize)
351         NormalizeRadialProfile(dst, radialSteps);
352 }
```

11.41.2.7 `template<typename T > T ComputeStdDev (T * data, int len)`

```

222 {
223     T sum = 0.0, sum2=0.0;
```

```

224     for (int a=0;a<len;a++) {
225         sum+=data[a];
226         sum2+=data[a]*data[a];
227     }
228     T mean = sum / len;
229     return sqrt(sum2 / len- mean * mean);
230 }
```

11.41.2.8 void dbgout(const std::string & s)

```

143             {
144     OutputDebugString(s.c_str());
145     printf(s.c_str());
146     WriteToLog(s.c_str());
147 }
```

11.41.2.9 void dbgprintf(const char * fmt, ...)

```

149             {
150     va_list ap;
151     va_start(ap, fmt);
152
153     char buf[512];
154     VSNPRINTF(buf, sizeof(buf), fmt, ap);
155     OutputDebugString(buf);
156     fputs(buf,stdout);
157     WriteToLog(buf);
158
159     va_end(ap);
160 }
```

11.41.2.10 void dbglogfile(const char * path)

```

50 {
51     logFilename = path;
52 }
```

11.41.2.11 template<typename T> void DeleteAllElems(T & c)

```

19             {
20     for(typename T::iterator i=c.begin();i!=c.end();++i)
21         delete *i;
22     c.clear();
23 }
```

11.41.2.12 template<typename T> T erf(T x)

```

374 {
375     // constants
376     T a1 = 0.254829592f;
377     T a2 = -0.284496736f;
378     T a3 = 1.421413741f;
379     T a4 = -1.453152027f;
380     T a5 = 1.061405429f;
381     T p = 0.3275911f;
382
383     // Save the sign of x
384     int sign = 1;
385     if (x < 0)
386         sign = -1;
387     x = fabs(x);
388
389     // A&S formula 7.1.26
390     T t = 1.0f/(1.0f + p*x);
391     T y = 1.0f - (((a5*t + a4)*t) + a3)*t + a2)*t + a1)*t*exp(-x*x);
392
393     return sign*y;
394 }
```

11.41.2.13 std::string file_ext(const char * f)

```

121     int l=strlen(f)-1;
122     while (l > 0) {
123         if (f[l] == '.')
124             return &f[l+1];
125         l--;
126     }
127     return "";
128 }
```

11.41.2.14 void FloatToJPEGFile(const char * name, const float * d, int w, int h)

```

190 {
191     uchar* zlut_bytes = floatToNormalizedInt(d, w,h, (
192         uchar)255);
193     WriteJPEGFile(zlut_bytes, w, h, name, 99);
194     delete[] zlut_bytes;
195 }
```

11.41.2.15 template<typename T> void floatToNormalizedInt(T * dst, const float * src, uint w, uint h, T maxValue)

```

200 {
201     float maxv = src[0];
202     float minv = src[0];
203     for (uint k=0;k<w*h;k++) {
204         maxv = std::max(maxv, src[k]);
205         minv = std::min(minv, src[k]);
206     }
207     for (uint k=0;k<w*h;k++)
208         dst[k] = maxValue * (src[k]-minv) / (maxv-minv);
209 }
```

11.41.2.16 template<typename T> T* floatToNormalizedInt(const float * src, uint w, uint h, T maxValue)

```

214 {
215     T* r = new T[w*h];
216     floatToNormalizedInt(r,src,w,h, maxValue);
217     return r;
218 }
```

11.41.2.17 void GenerateGaussianSpotImage(ImageData * img, vector2f pos, float sigma, float I0, float Ibg)

```

426 {
427     float edenom = 1/sqrt(2*sigma*sigma);
428     for (int y=0;y<img->h;y++) {
429         for(int x=0;x<img->w;x++) {
430             float DeltaX = 0.5f * erf( (x-pos.x + .5f) * edenom ) - 0.5f *
431                 erf((x-pos.x - .5f) * edenom);
432             float DeltaY = 0.5f * erf( (y-pos.y + .5f) * edenom ) - 0.5f *
433                 erf((y-pos.y - .5f) * edenom);
434             img->at(x,y) = Ibg + I0 * DeltaX * DeltaY;
435         }
436     }
437 }
```

11.41.2.18 void GenerateImageFromLUT (**ImageData * *image*, **ImageData** * *zlut*, float *minradius*, float *maxradius*, vector3f *pos*, bool *useSplineInterp* = true, int *ovs* = 4)**

```

358 {
359 // lut.w = radialcov * ( (image->w/2 * roicov) - minradius );
360 // lut.w = radialcov * ( maxradius - minradius );
361
362 float radialcov = zlut->w / (maxradius-minradius);
363 float* zinterp = (float*)ALLOCA(zlut->w * sizeof(float));
364
365 if (splineInterp) {
366     int iz = std::max(1, std::min(zlut->h-3, (int)pos.z));
367     float weights[4];
368     float fz = pos.z-iz;
369     ComputeBSplineWeights(weights, fz);
370     // Interpolate ZLUT using B-spline weights
371     for (int r=0;r<zlut->w;r++) {
372         float zlutv = 0;
373         for (int i=0;i<4;i++)
374             zlutv += weights[i] * zlut->at(r, i-1+iz);
375         zinterp[r] = zlutv;
376     }
377 }
378 else {
379     // The two Z planes to interpolate between
380     int iz = (int)pos.z;
381     if (iz < 0)
382         zinterp = zlut->data;
383     else if (iz>=zlut->h-1)
384         zinterp = &zlut->data[ (zlut->h-1)*zlut->w ];
385     else {
386         float* zlut0 = &zlut->data [ (int)pos.z * zlut->w ];
387         float* zlut1 = &zlut->data [ ((int)pos.z + 1) * zlut->w ];
388         zinterp = (float*)ALLOCA(sizeof(float)*zlut->w);
389         for (int r=0;r<zlut->w;r++)
390             zinterp[r] = Lerp(zlut0[r], zlut1[r], pos.z-iz);
391     }
392 }
393
394 int oversampleWidth=oversampleSubdiv, oversampleHeight=oversampleSubdiv;
395 float oxstep = 1.0f / oversampleWidth;
396 float oystep = 1.0f / oversampleHeight;
397
398 for (int y=0;y<image->h;y++)
399     for (int x=0;x<image->w;x++)
400     {
401         float s = 0.0f;
402
403         for (int ox=0;ox<oversampleWidth;ox++)
404             for (int oy=0;oy<oversampleHeight;oy++) {
405
406                 float X = x+(ox+0.5f)*oxstep - pos.x - 0.5f;
407                 float Y = y+(oy+0.5f)*oystep - pos.y - 0.5f;
408
409                 float pixr = sqrtf(X*X+Y*Y);
410                 float r = (pixr - minradius) * radialcov;
411
412                 if (r > zlut->w-2)
413                     r = zlut->w-2;
414                 if (r < 0) r = 0;
415
416                 int i=(int)r;
417                 s += Lerp(zinterp[i], zinterp[i+1], r-i);
418             }
419
420             image->at(x,y) = s/(oversampleWidth*oversampleHeight);
421     }
422 }
```

11.41.2.19 void GenerateTestImage (**ImageData & *img*, float *xp*, float *yp*, float *size*, float *MaxPhotons*)**

```

163 {
164     float S = 1.0f/sqrt(size);
165     for (int y=0;y<img.h;y++) {
166         for (int x=0;x<img.w;x++) {
167             float X = x - xp;// + 0.5;
168             float Y = y - yp;// + 0.5;
169             float r = sqrtf(X*X+Y*Y)+1;
170             float v = sinf(r/(5*S)) * expf(-r*r*S*0.001f);
171             img.at(x,y)=v;
```

```

172         }
173     }
174
175     if (SNratio>0) {
176         ApplyGaussianNoise(img, 1.0f/SNratio);
177     }
178     img.normalize();
179 }
```

11.41.2.20 std::string GetCurrentOutputPath (bool ext = true)

```

20 {
21     std::string base = "D:\\TestImages\\TestOutput\\";
22     std::string search = base + "*";
23     LPCSTR w_folder = _T(search.c_str());
24
25     WIN32_FIND_DATA FindFileData;
26     HANDLE hFind;
27
28     hFind = FindFirstFile(w_folder,&FindFileData);
29     std::string dirName;
30     while(FindNextFile(hFind,&FindFileData))
31         dirName = FindFileData.cFileName;
32     if(ext)
33         return SPrintf("%s%s\\%s",base.c_str(),dirName.c_str(),"ZLUTDiag\\");
34     else
35         return SPrintf("%s%s",base.c_str(),dirName.c_str());
36 }
```

11.41.2.21 std::string GetDirectoryFromPath (std::string fullPath)

```

105 {
106     for (int i=fullpath.size()-1;i>=0;i--) {
107         if (fullpath[i] == '/' || fullpath[i] == '\\') {
108             return fullname.substr(0, i);
109         }
110     }
111     return "";
112 }
```

11.41.2.22 void GetFormattedTimeString (char * output)

```

39 {
40     time_t rawtime;
41     struct tm * timeinfo;
42     time(&rawtime);
43     timeinfo = localtime(&rawtime);
44     sprintf(output, "%02d%02d%02d-%02d%02d%02d", timeinfo->tm_year-100, timeinfo->tm_mon+1, timeinfo->tm_mday,
45     timeinfo->tm_hour, timeinfo->tm_min, timeinfo->tm_sec);
46 }
```

11.41.2.23 std::string GetLocalModuleFilename ()

```

55 {
56 #ifdef WIN32
57     char path[256];
58     HMODULE hm = NULL;
59
60     GetModuleHandleExA(GET_MODULE_HANDLE_EX_FLAG_FROM_ADDRESS |
61                         GET_MODULE_HANDLE_EX_FLAG_UNCHANGED_REFCOUNT,
62                         (LPCSTR) &GetLocalModuleFilename, &hm);
63
64     GetModuleFileNameA(hm, path, sizeof(path));
65     return path;
66 #else
67     #error GetLocalModuleName() not implemented for this platform
68 #endif
69 }
```

11.41.2.24 std::string GetLocalModulePath()

```
115 {
116     std::string dllpath = GetLocalModuleFilename();
117     return GetDirectoryFromPath(dllpath);
118 }
```

11.41.2.25 double GetPreciseTime()

```
680 {
681     uint64_t freq, time;
682     QueryPerformanceCounter((LARGE_INTEGER*)&time);
683     QueryPerformanceFrequency((LARGE_INTEGER*)&freq);
685     return (double)time / (double)freq;
687 }
```

11.41.2.26 template<typename T > T Interpolate(T * image, int width, int height, float x, float y, bool * outside = 0) [inline]

```
44 {
45     int rx=x, ry=y;
46     if (rx<0 || ry <0 || rx+1 >= width || ry+1>=height) {
47         if (outside) *outside=true;
48         return 0.0f;
49     }
50     if (outside) *outside=false;
51
52     T v00 = image[width*ry+rx];
53     T v10 = image[width*ry+rx+1];
54     T v01 = image[width*(ry+1)+rx];
55     T v11 = image[width*(ry+1)+rx+1];
56
57     T v0 = Lerp(v00, v10, x-rx);
58     T v1 = Lerp(v01, v11, x-rx);
59
60     return Lerp(v0, v1, y-ry);
61 }
```

11.41.2.27 template<typename T > T Interpolate1D(T * d, int len, float x) [inline]

```
65 {
66     int fx = (int)x;
67     if (fx < 0) return d[0];
68     if (fx >= len-1) return d[len-1];
69     return (d[fx+1]-d[fx]) * (x-fx) + d[fx];
70 }
```

11.41.2.28 template<typename T > T Interpolate1D(const std::vector<T> & d, float x) [inline]

```
74 {
75     return Interpolate1D(&d[0],d.size(),x);
76 }
```

11.41.2.29 template<typename T > bool isNaN(const T & v)

```
6
7     return !(v == v);
8 }
```

11.41.2.30 template<typename T> T Lerp (T a, T b, float x) [inline]

```
40 { return a + (b-a)*x; }
```

11.41.2.31 int NearestPowerOf2 (int v)

```
693 {
694     int r=1;
695     while (r < v)
696         r *= 2;
697     if ( fabsf(r-v) < fabsf(r/2-v) )
698         return r;
699     return r/2;
700 }
```

11.41.2.32 int NearestPowerOf3 (int v)

```
703 {
704     int r=1;
705     while (r < v)
706         r *= 3;
707     if ( fabsf(r-v) < fabsf(r/3-v) )
708         return r;
709     return r/3;
710 }
```

11.41.2.33 template<typename TPixel> void normalize (TPixel * d, uint w, uint h)

```
28 {
29     TPixel maxv = d[0];
30     TPixel minv = d[0];
31     for (uint k=0;k<w*h;k++) {
32         maxv = std::max(maxv, d[k]);
33         minv = std::min(minv, d[k]);
34     }
35     for (uint k=0;k<w*h;k++)
36         d[k]=(d[k]-minv)/(maxv-minv);
37 }
```

11.41.2.34 void NormalizeRadialProfile (float * prof, int rsteps)

```
260 {
261     double sum=0.0f;
262     for (int i=0;i<rsteps;i++)
263         sum += prof[i];
264
265     float mean =sum/rsteps;
266     double rmssum2 = 0.0;
267
268     for (int i=0;i<rsteps;i++) {
269         prof[i] -= mean;
270         rmssum2 += prof[i]*prof[i];
271     }
272     double invTotalrms = 1.0f/sqrt(rmssum2/rsteps);
273     for (int i=0;i<rsteps;i++)
274         prof[i] *= invTotalrms;
275
276 /*
277     scalar_t minValue = prof[0];
278     for (int i=0;i<rsteps;i++)
279         if(prof[i]<minValue) minValue=prof[i];
280
281     float rms=0;
282     for (int i=0;i<rsteps;i++) {
283         prof[i]-=minValue;
284         rms += prof[i]*prof[i];
285     }
286     rms=1.0f/sqrt(rms);
287     for (int i=0;i<rsteps;i++) {
288         prof[i]*=rms;
289     */
290 }
```

11.41.2.35 void NormalizeZLUT (float * *zlut*, int *numLUTs*, int *planes*, int *radialsteps*)

```

294 {
295     for(int i=0;i<numBeads;i++)
296         for (int j=0;j<planes;j++)
297             NormalizeRadialProfile(&zlut[radialsteps*planes*i + radialsteps*j],
298             radialsteps);
298 }
```

11.41.2.36 template<typename T > T qselect (T * *data*, int *start*, int *end*, int *k*)

```

234 {
235     if (end-start==1)
236         return data[start];
237
238     // select one of the elements as pivot
239     int p = 0;
240     T value = data[p+start];
241     // swap with last value
242     std::swap(data[p+start], data[end-1]);
243
244     // move all items < pivot to the left
245     int nSmallerItems=0;
246     for(int i=start;i<end-1;i++)
247         if(data[i]<value) {
248             std::swap(data[i], data[start+nSmallerItems]);
249             nSmallerItems++;
250         }
251     // pivot is now at [# items < pivot]
252     std::swap(data[start+nSmallerItems], data[end-1]);
253
254     // we are trying to find the kth element
255     // so if pivotpos == k, we found it
256     // if k < pivotpos, we need to recurse left side
257     // if k > pivotpos, we need to recurse right side
258     int pivotpos = start+nSmallerItems;
259     if (k == pivotpos)
260         return data[k];
261     else if (k < pivotpos)
262         return qselect(data, start, pivotpos, k);
263     else
264         return qselect(data, pivotpos+1, end, k);
265 }
```

11.41.2.37 std::vector< std::vector<float> > ReadCSV (const char * *filename*, char *sep* = '\t')

```

468 {
469     std::list< std::vector <float> > data;
470
471     FILE *f=fopen(filename,"r");
472     if (f) {
473         std::string buf;
474         std::vector<float> vals;
475         while (!feof(f)) {
476             char c=fgetc(f);
477             if (c == sep || c=='\n') {
478                 vals.push_back( atof(buf.c_str()) );
479                 buf.clear();
480             }
481             if (c == '\n') {
482                 data.push_back( vals );
483                 vals.clear();
484             }
485             if (c != sep && c!='\n')
486                 buf+=c;
487         }
488         fclose(f);
489     }
490
491     std::vector<std::vector<float> > r;
492     r.reserve(data.size());
493     r.insert(r.begin(), data.begin(), data.end());
494     return r;
495 }
496 }
```

11.41.2.38 `ImageData ReadJPEGFile(const char * fn)`

```

634 {
635     int w, h;
636     uchar* imgdata;
637     std::vector<uchar> jpgdata = ReadToByteBuffer(fn);
638     ReadJPEGFile(&jpgdata[0], jpgdata.size(), &imgdata, &w,&h);
639
640     float* fbuf = new float[w*h];
641     for (int x=0;x<w*h;x++)
642         fbuf[x] = imgdata[x]/255.0f;
643     delete[] imgdata;
644
645     return ImageData(fbuf,w,h);
646 }
```

11.41.2.39 `int ReadJPEGFile(uchar * srcbuf, int srclen, uchar ** data, int * width, int * height)`

```

13 {
14     struct jpeg_decompress_struct cinfo;
15
16     JSAMPARRAY buffer;          /* Output row buffer */
17     int row_stride;           /* physical row width in output buffer */
18     my_error_mgr jerr;
19     cinfo.err = jpeg_std_error(&jerr.pub);
20     jpeg_create_decompress(&cinfo);
21
22     j_mem_src(&cinfo, srcbuf, srclen);
23
24     /* Step 3: read file parameters with jpeg_read_header() */
25     jpeg_read_header(&cinfo, TRUE);
26
27     jpeg_start_decompress(&cinfo);
28     row_stride = cinfo.output_width * cinfo.output_components;
29     /* Make a one-row-high sample array that will go away when done with image */
30     buffer = (*cinfo.mem->alloc_sarray) ((j_common_ptr) &cinfo, JPOOL_IMAGE, row_stride, 1);
31
32     *width = cinfo.output_width;
33     *height = cinfo.output_height;
34     *data = new uchar[cinfo.output_width*cinfo.output_height];
35
36 //  ResizeLVAarray2D(output, cinfo.output_height, cinfo.output_width);
37
38     /* Step 6: while (scan lines remain to be read) */
39     /*          jpeg_read_scanlines(...); */
40
41     /* Here we use the library's state variable cinfo.output_scanline as the
42      * loop counter, so that we don't have to keep track ourselves.
43      */
44     uchar* dst = *data;
45     while (cinfo.output_scanline < cinfo.output_height) {
46         /* jpeg_read_scanlines expects an array of pointers to scanlines.
47         * Here the array is only one element long, but you could ask for
48         * more than one scanline at a time if that's more convenient.
49         */
50         jpeg_read_scanlines(&cinfo, buffer, 1);
51         /* Assume put_scanline_someplace wants a pointer and sample count. */
52
53         unsigned char* src = buffer[0];
54         if (cinfo.output_components == 1) {
55             memcpy(dst, src, cinfo.output_width);
56         } else {
57             for (uint x=0;x<cinfo.output_width;x++)
58                 dst[x] = src[x * cinfo.output_components];
59         }
60         dst += cinfo.output_width;
61     }
62
63     /* Step 7: Finish decompression */
64     jpeg_finish_decompress(&cinfo);
65     /* We can ignore the return value since suspension is not possible
66      * with the stdio data source.
67      */
68
69     /* Step 8: Release JPEG decompression object */
70
71     /* This is an important step since it will release a good deal of memory. */
72     jpeg_destroy_decompress(&cinfo);
73
74     /* After finish_decompress, we can close the input file.
75      * Here we postpone it until after no more JPEG errors are possible,
76      */
77 }
```

```

76     * so as to simplify the setjmp error logic above. (Actually, I don't
77     * think that jpeg_destroy can do an error exit, but why assume anything...)
78     */
79 //  fclose(infile);
80
81 /* At this point you may want to check to see whether any corrupt-data
82  * warnings occurred (test whether jerr.pub.num_warnings is nonzero).
83  */
84
85 return 1;
86 }

```

11.41.2.40 `ImageData ReadLUTFile(const char * lutfile)`

```

729 {
730     PathSeparator sep(lutfile);
731     if(sep.extension == "jpg") {
732         return ReadJPEGFile(lutfile);
733     }
734     else {
735         std::string fn = lutfile;
736         fn = std::string(fn.begin(), fn.begin() + fn.find('#'));
737         std::string num( ++(sep.extension.begin() + sep.extension.find('#')), sep.extension.end());
738         int lutIndex = atoi(num.c_str());
739
740         int nbeads, nplanes, nsteps;
741         FILE *f = fopen(fn.c_str(), "rb");
742
743         if (!f)
744             throw std::runtime_error("Can't open " + fn);
745
746         fread(&nbeads, 4, 1, f);
747         fread(&nplanes, 4, 1, f);
748         fread(&nsteps, 4, 1, f);
749
750
751         fseek(f, 12 + 4 * (nsteps*nplanes * lutIndex), SEEK_SET);
752         ImageData lut = ImageData::alloc(nsteps,nplanes);
753         fread(lut.data, 4, nsteps*nplanes,f);
754         fclose(f);
755         lut.normalize();
756         return lut;
757     }
758 }

```

11.41.2.41 `std::vector<uchar> ReadToByteBuffer(const char * filename)`

```

615 {
616     FILE *f = fopen(filename, "rb");
617
618     if (!f)
619         throw std::runtime_error(Sprintf("%s was not found", filename));
620
621     fseek(f, 0, SEEK_END);
622     int len = ftell(f);
623     fseek(f, 0, SEEK_SET);
624
625     std::vector<uchar> buf(len);
626     fread(&buf[0], 1, len, f);
627
628     fclose(f);
629     return buf;
630 }

```

11.41.2.42 `std::vector<vector3f> ReadVector3CSV(const char * file, char sep = '\t')`

```

500 {
501     auto data=ReadCSV(file ,sep);
502
503     std::vector<vector3f> r(data.size());
504
505     for (int i=0;i<data.size();i++){
506         r[i]=vector3f(data[i][0],data[i][1],data[i][2]);
507     }
508     return r;
509 }

```

11.41.2.43 std::string SPrintf(const char * fmt, ...)

```

132             {
133     va_list ap;
134     va_start(ap, fmt);
135
136     char buf[512];
137     VSNPRINTF(buf, sizeof(buf), fmt, ap);
138
139     va_end(ap);
140     return buf;
141 }
```

11.41.2.44 template<typename T > T StdDeviation (T * start, T * end)

```

139             {
140     T sum=0,sum2=0;
141     for (T* s = start; s!=end; ++s) {
142         sum+=*s; sum2+=(*s)*(s);
143     }
144
145     T invN = 1.0f/(end-start);
146     T mean = sum * invN;
147     return sqrt(sum2 * invN - mean * mean);
148 }
```

11.41.2.45 void WriteArrayAsCSVRow (const char * file, float * d, int len, bool append)

```

543 {
544     FILE *f = fopen(file, append?"a":"w");
545     if(f) {
546         for (int i=0;i<len;i++)
547             fprintf(f, "%.7f\t", d[i]);
548
549         fprintf(f, "\n");
550         fclose(f);
551     }
552     else
553         dbgprintf("WriteArrayAsCSVRow: Unable to open file %s\n", file);
554 }
```

11.41.2.46 void WriteComplexImageAsCSV (const char * file, std::complex< float > * d, int w, int h, const char * labels[] = 0)

```

586 {
587     FILE* f = fopen(file, "w");
588
589     if (!f) {
590         dbgprintf("WriteComplexImageAsCSV: Unable to open file %s\n", file);
591         return;
592     }
593
594     if (labels) {
595         for (int i=0;i<w;i++) {
596             fprintf(f, "%s;\t", labels[i]);
597         }
598         fputs("\n", f);
599     }
600
601     for (int y=0;y<h;y++) {
602         for (int x=0;x<w;x++)
603         {
604             float i=d[y*w+x].imag();
605             fprintf(f, "%f%+fi", d[y*w+x].real(), i);
606             if(x<w-1) fputs("\t", f);
607         }
608         fprintf(f, "\n");
609     }
610
611     fclose(f);
612 }
```

11.41.2.47 void WriteImageAsCSV (const char * file, float * d, int w, int h, const char * labels[] = 0)

```

557 {
558     FILE* f = fopen(file, "w");
559
560     if (f) {
561
562         if (labels) {
563             for (int i=0;i<w;i++) {
564                 fprintf(f, "%s;\t", labels[i]);
565             }
566             fputs("\n", f);
567         }
568
569         for (int y=0;y<h;y++) {
570             for (int x=0;x<w;x++) {
571                 fprintf(f, "% .10f", d[y*w+x]);
572                 if(x<w-1) fputs("\t", f);
573             }
574             fprintf(f, "\n");
575         }
576     }
577
578     fclose(f);
579 }
580 else
581     dbgprintf("WriteImageAsCSV: Unable to open file %s\n", file);
582 }
```

11.41.2.48 void WriteJPEGFile (uchar * data, int w, int h, const char * filename, int quality)

```

90 {
91     /* This struct contains the JPEG compression parameters and pointers to
92      * working space (which is allocated as needed by the JPEG library).
93      * It is possible to have several such structures, representing multiple
94      * compression/decompression processes, in existence at once. We refer
95      * to any one struct (and its associated working data) as a "JPEG object".
96      */
97     struct jpeg_compress_struct cinfo;
98     /* This struct represents a JPEG error handler. It is declared separately
99      * because applications often want to supply a specialized error handler
100     * (see the second half of this file for an example). But here we just
101     * take the easy way out and use the standard error handler, which will
102     * print a message on stderr and call exit() if compression fails.
103     * Note that this struct must live as long as the main JPEG parameter
104     * struct, to avoid dangling-pointer problems.
105     */
106    struct jpeg_error_mgr jerr;
107    /* More stuff */
108    JSAMPROW row_pointer[1]; /* pointer to JSAMPLE row[s] */
109    int row_stride; /* physical row width in image buffer */
110
111    /* Step 1: allocate and initialize JPEG compression object */
112
113    /* We have to set up the error handler first, in case the initialization
114     * step fails. (Unlikely, but it could happen if you are out of memory.)
115     * This routine fills in the contents of struct jerr, and returns jerr's
116     * address which we place into the link field in cinfo.
117     */
118    cinfo.err = jpeg_std_error(&jerr);
119    /* Now we can initialize the JPEG compression object. */
120    jpeg_create_compress(&cinfo);
121
122    /* Step 2: specify data destination (eg, a file) */
123    /* Note: steps 2 and 3 can be done in either order. */
124
125    uint len = w*h * 2;
126    uchar* memBuf = new uchar[len];
127    j_mem_dest(&cinfo, (void**)&memBuf, &len);
128    //jpeg_stdio_dest(&cinfo, outfile);
129
130    /* Step 3: set parameters for compression */
131
132    /* First we supply a description of the input image.
133     * Four fields of the cinfo struct must be filled in:
134     */
135    cinfo.image_width = w; /* image width and height, in pixels */
136    cinfo.image_height = h;
137    cinfo.input_components = 1; /* # of color components per pixel */
138    cinfo.in_color_space = JCS_GRAYSCALE; /* colorspace of input image */
139    /* Now use the library's routine to set default compression parameters.
```

```

140     * (You must set at least cinfo.in_color_space before calling this,
141     * since the defaults depend on the source color space.)
142     */
143 jpeg_set_defaults(&cinfo);
144 /* Now you can set any non-default parameters you wish to.
145  * Here we just illustrate the use of quality (quantization table) scaling:
146  */
147 jpeg_set_quality(&cinfo, quality, TRUE /* limit to baseline-JPEG values */);
148
149 /* Step 4: Start compressor */
150
151 /* TRUE ensures that we will write a complete interchange-JPEG file.
152  * Pass TRUE unless you are very sure of what you're doing.
153  */
154 jpeg_start_compress(&cinfo, TRUE);
155
156 /* Step 5: while (scan lines remain to be written) */
157 /*      jpeg_write_scanlines(...); */
158
159 /* Here we use the library's state variable cinfo.next_scanline as the
160  * loop counter, so that we don't have to keep track ourselves.
161  * To keep things simple, we pass one scanline per call; you can pass
162  * more if you wish, though.
163  */
164 row_stride = w;    /* JSAMPLEs per row in image_buffer */
165
166 while (cinfo.next_scanline < cinfo.image_height) {
167     /* jpeg_write_scanlines expects an array of pointers to scanlines.
168      * Here the array is only one element long, but you could pass
169      * more than one scanline at a time if that's more convenient.
170      */
171     row_pointer[0] = &data[cinfo.next_scanline * row_stride];
172     (void) jpeg_write_scanlines(&cinfo, row_pointer, 1);
173 }
174 jpeg_finish_compress(&cinfo);
175 jpeg_destroy_compress(&cinfo);
176
177
178 FILE *f = fopen(filename, "wb");
179 if (f) {
180     fwrite(memBuf, 1, len, f);
181     fclose(f);
182 } else
183     dbgprintf("Failed to open file %s for writing\n", filename);
184
185 delete[] memBuf;
186 }
```

11.41.2.49 void WriteJPEGFile(const char * name, const ImageData & img) [inline]

```
190 { FloatToJPEGFile(name, img.data, img.w,img.h); }
```

11.41.2.50 void WriteTrace(std::string file, vector3f * results, int nResults)

```

513 {
514     FILE *f = fopen(filename.c_str(), "w");
515
516     if (!f)
517         throw std::runtime_error(SPrintf("Can't open %s", filename.c_str()));
518 }
519
520     for (int i=0;i<nResults;i++)
521     {
522         fprintf(f, "%.7f\t%.7f\t%.7f\n", results[i].x, results[i].y, results[i].z);
523     }
524
525     fclose(f);
526 }
```

11.41.2.51 void WriteVectorAsCSVRow (const char * file, std::vector< float > d, bool append)

```

529 {
530     FILE *f = fopen(file, append?"a":"w");
531     if(f) {
532         for (int i=0;i<d.size();i++)
533             fprintf(f, "%1.7f\t", d[i]);
534         fprintf(f, "\n");
535         fclose(f);
536     }
537 } else
538     dbgprintf("WriteArrayAsCSVRow: Unable to open file %s\n", file);
540 }
```

11.42 cudatrack-test/test.cu File Reference

```

#include "cuda_runtime.h"
#include "device_launch_parameters.h"
#include "std_incl.h"
#include "utils.h"
#include <cassert>
#include <cstdlib>
#include <stdio.h>
#include <windows.h>
#include <cstdarg>
#include <valarray>
#include "random_distr.h"
#include <stdint.h>
#include "gpu_utils.h"
#include "QueuedCUDATracker.h"
#include "QueuedCPUTracker.h"
#include "../cputrack-test/SharedTests.h"
#include "BenchmarkLUT.h"
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/generate.h>
#include <thrust/reduce.h>
#include <thrust/functional.h>
#include "FisherMatrix.h"
#include "testutils.h"
#include "ResultManager.h"
#include "ExtractBeadImages.h"
```

Classes

- struct [SpeedInfo](#)

Functions

- void [BenchmarkParams \(\)](#)
- std::string [getPath \(const char *file\)](#)
- [__device__ float2 mul_conjugate \(float2 a, float2 b\)](#)
- void [ShowCUDAError \(\)](#)
- [__device__ float compute \(int idx, float *buf, int s\)](#)

- `__global__ void testWithGlobal (int n, int s, float *result, float *buf)`
- `__global__ void testWithShared (int n, int s, float *result)`
- `void TestSharedMem ()`
- `void QTrkCompareTest ()`
- `void listDevices ()`
- `__global__ void SimpleKernel (int N, float *a)`
- `void TestAsync ()`
- `__global__ void emptyKernel ()`
- `float SpeedTest (const QTrkSettings &cfg, QueuedTracker *qtrk, int count, bool haveZLUT, LocMode_t locType, float *scheduleTime, bool gainCorrection=false)`
- `int NearestPowerOfTwo (int v)`
- `int SmallestPowerOfTwo (int minval)`
- `SpeedInfo SpeedCompareTest (int w, LocalizeModeEnum locMode, bool haveZLUT, int qi_iterations=5)`
- `void ProfileSpeedVsROI (LocalizeModeEnum locMode, const char *outputcsv, bool haveZLUT, int qi_iterations)`
- `void CompareAccuracy (const char *lutfile)`
- `void BasicQTrkTest ()`
- `void BasicQTrkTest_RM ()`
- `void TestGauss2D (bool calib)`
- `void TestRadialLUTGradientMethod ()`
- `void QICompare (const char *lutfile)`
- `void TestBenchmarkLUT ()`
- template<typename T>
 `void check_arg (const std::vector< std::string > &args, const char *name, T *param)`
- `void check_strarg (const std::vector< std::string > &args, const char *name, std::string *param)`
- `int CmdLineRun (int argc, char *argv[])`
- `void BuildZLUT (std::string folder, outputter *output)`
- `int main (int argc, char *argv[])`

Variables

- `__shared__ float cudaSharedMem []`
- `std::vector< float > cmp_cpu_qi_prof`
- `std::vector< float > cmp_gpu_qi_prof`
- `std::vector< std::complex< float > > cmp_cpu_qi_fft_out`
- `std::vector< std::complex< float > > cmp_gpu_qi_fft_out`

11.42.1 Function Documentation

11.42.1.1 void BasicQTrkTest()

```

536 {
537     QTrkComputedConfig cc;
538     cc.width = cc.height = 100;
539     cc.Update();
540     QueuedCUDATracker qtrk(cc);
541
542     float zmin=1,zmax=5;
543     ImageData img = ImageData::alloc(cc.width,cc.
544     height);
545     float pos_x = cc.width/2 - 5;
546     float pos_y = cc.height/2 + 3;
547     GenerateTestImage(img, pos_x, pos_y, (zmin+zmax)/2, 0);
548
549     int N = 100000;
550 #ifdef _DEBUG
551     N = 10000;
552 #endif

```

```

553     double t = GetPreciseTime();
554     qtrk.SetLocalizationMode((LocMode_t)(LT_QI |
555     LT_NormalizeProfile));
556     for (int i=0;i<N;i++)
557     {
558         LocalizationJob job ( i, 0, 0, 0);
559         qtrk.ScheduleLocalization((uchar*)img.data, sizeof(float)*cc.
560         width, QTrkFloat, &job);
561         if(i%std::max(1,(int)(N*0.1))==0) dbgprintf("Queued: %d / %d\n", i, N);
562     }
563     WaitForFinish(&qtrk, N);
564     t = GetPreciseTime() - t;
565     dbgprintf("Speed: %d imgs/s (Only QI, %d iterations)\n", (int)(N / t), cc.
566     qi_iterations);
567     int count = 0;
568
569     while(qtrk.GetResultCount() != 0){
570         LocalizationResult res;
571         qtrk.FetchResults(&res,1);
572         if( res.pos.x > pos_x + 0.01f || res.pos.x < pos_x - 0.01f || res.
573         pos.y > pos_y + 0.01f || res.pos.y < pos_y - 0.01f ){
574             if(count < 100)
575                 dbgprintf("Location frame %d: (%02f,%02f)\n",res.job.
576                 frame, res.pos.x, res.pos.y);
577             count++;
578         }
579     }
580     dbgprintf("Errors: %d/%d (%f%%)\n", count, N, (float)100*count/N);
581     img.free();
582 }
```

11.42.1.2 void BasicQTrkTest_RM()

```

581 {
582     QTrkComputedConfig cc;
583     //cc.qi_iterations = 10;
584     cc.width = cc.height = 100;
585     cc.Update();
586     QueuedCUDATracker qtrk(cc);
587
588     float zmin=1,zmax=5;
589     ImageData img = ImageData::alloc(cc.width,cc.
590     height);
591
592     // Positions to set
593     float pos_x = cc.width/2 - 5;
594     float pos_y = cc.height/2 + 3;
595     GenerateTestImage(img, pos_x, pos_y, (zmin+zmax)/2, 0);
596
597     int N = 100000;
598 #ifdef _DEBUG
599     N = 100000;
600 #endif
601     qtrk.SetLocalizationMode((LocMode_t)(LT_QI |
602     LT_NormalizeProfile));
603
604     ResultManagerConfig RMcfg;
605     RMcfg.numBeads = 1;
606     RMcfg.numFrameInfoColumns = 0;
607     RMcfg.scaling = vector3f(1.0f,1.0f,1.0f);
608     RMcfg.offset = vector3f(0.0f,0.0f,0.0f);
609     RMcfg.writeInterval = 4000;
610     RMcfg.maxFramesInMemory = 0;
611     RMcfg.binaryOutput = false;
612
613     std::vector<std::string> colnames;
614     for(int ii = 0;ii<RMcfg.numFrameInfoColumns;ii++){
615         colnames.push_back(SPrintf("%d",ii));
616     }
617
618     outputter output(Files+Images);
619
620     ResultManager RM(
621         SPrintf("%s\\RMOutput.txt",output.folder.c_str()).c_str(),
622         SPrintf("%s\\RMFrameInfo.txt",output.folder.c_str()).c_str(),
623         &RMcfg, colnames);
624
625     RM.SetTracker(&qtrk);
626     double t = GetPreciseTime();
627     for (int i=0;i<N;i++)
628     {
```

```

627     LocalizationJob job ( i, 0, 0, 0);
628     qtrk.ScheduleLocalization((uchar*)img.data, sizeof(float)*cc.
629     width, QTrkFloat, &job);
630     //if(i%std::max(1,N/1000)==0) dbgprintf("Queued: %d / %d\n", i, N);
631     }
632     printf("\nDone queueing!\n");
633     // Tell the tracker to perform the localizations left in the queue regardless of batchSize
634     qtrk.Flush();
635     // Halt the test (=timer) until all localizations are done.
636     while(RM.GetFrameCounters().localizationsDone < N);
637     t = GetPreciseTime() - t;
638
639     // Tell the resultmanager to print the final available results regardless of writeInterval
640     RM.Flush();
641     while(RM.GetFrameCounters().lastSaveFrame != N);
642
643     dbgprintf("Speed: %d imgs/s (Only QI, %d iterations)\n", (int)(N / t), cc.
644     qi_iterations);
645     img.free();
646 }
```

11.42.1.3 void BenchmarkParams()

```

121 {
122     /*
123     - Accuracy vs ROIsize
124     - Speed vs ROIsize
125     */
126 #ifdef _DEBUG
127     int n = 50;
128 #else
129     int n = 300;
130 #endif
131
132     int mpv = 10000;
133     float pixel_size = 120, lutstep = 50;
134
135     for (int zlutbias=0;zlutbias<2;zlutbias++) {
136         float range_in_nm=0;
137         for (int bias=0;bias<2;bias++) {
138             for (int i=0;i<5;i++)
139                 BenchmarkROISizes(SPrintf("roi_qi%d_bias%d_zlutbias%d.txt",i,bias,
zlutbias).c_str(), n, mpv, i, 0, range_in_nm, pixel_size, lutstep, zlutbias ?
BUILDLUT_BIASCORRECT : 0);
140             // for (int i=0;i<5;i++)
141             // BenchmarkROISizes(SPrintf("roi_qi%d_bias%d_wz.txt",i,bias).c_str(), n, mpv, i,
LT_LocalizeZWeighted, range_in_nm, pixel_size, lutstep);
142             BenchmarkROISizes( SPrintf("roi_xcor_bias%d_zlutbias%d.txt", bias,
zlutbias).c_str(), n, mpv, 0, LT_XCor1D, range_in_nm, pixel_size, lutstep, zlutbias ?
BUILDLUT_BIASCORRECT : 0);
143             // BenchmarkROISizes( SPrintf("roi_xcor_bias%d_wz.txt",bias).c_str(), n, mpv, 0, LT_XCor1D |
LT_LocalizeZWeighted, range_in_nm, pixel_size, lutstep);
144             range_in_nm = 200;
145         }
146     }
147
148     QTrkSettings basecfg;
149     basecfg.width = 80;
150     basecfg.height = 80;
151     basecfg.qi_iterations = 4;
152     basecfg.qi_roi_coverage = 1;
153     basecfg.qi_minradius=1;
154     basecfg.zlut_minradius=1;
155     basecfg.qi_radial_coverage = 2.5f;
156     basecfg.qi_angular_coverage = 0.7f;
157     basecfg.zlut_roi_coverage = 1;
158     basecfg.zlut_radial_coverage = 1.5f;
159     basecfg.com_bgcorrection = 0;
160     basecfg.qi_angstep_factor = 1.1f;
161     basecfg.zlut_angular_coverage = 0.7f;
162
163     //BenchmarkConfigParamRange (20000, &QTrkSettings::qi_iterations, &basecfg, linspace(1, 6, 6),
"qi_iterations_noise", mpv);
164 /*
165     BenchmarkConfigParamRange (n, &QTrkSettings::qi_radial_coverage, &basecfg, linspace(0.2f, 4.0f, 20),
"qi_rad_cov_noise", mpv );
166     BenchmarkConfigParamRange (n, &QTrkSettings::zlut_radial_coverage, &basecfg, linspace(0.2f, 4.0f, 20),
"zlut_rad_cov_noise", mpv);
167     BenchmarkConfigParamRange (n, &QTrkSettings::qi_iterations, &basecfg, linspace(1, 6, 6),
"qi_iterations_noise", mpv);
```

```

168     BenchmarkZAccuracy("zpos-noise.txt", n, mpv);
169
170     BenchmarkROISizes("roi-sizes.txt", n, 0);
171     BenchmarkConfigParamRange (n, &QTrkSettings::qi_radial_coverage, &basecfg, linspace(0.2f, 4.0f, 20),
172     "qi_rad_cov", 0);
173     BenchmarkConfigParamRange (n, &QTrkSettings::zlut_radial_coverage, &basecfg, linspace(0.2f, 4.0f, 20),
174     "zlut_rad_cov", 0);
175     BenchmarkConfigParamRange (n, &QTrkSettings::qi_iterations, &basecfg, linspace(1, 6, 6),
176     "qi_iterations", 0);
177     BenchmarkZAccuracy("zpos.txt", n, 0);*/
178 }
```

11.42.1.4 void BuildZLUT (std::string *folder*, outputter * *output*)

```

945 {
946     int ROISize = 100;
947     std::vector<BeadPos> beads = read_beadlist(SPrintf("%sbeadlist.txt", folder.c_str()));
948
949
950     int numImgInStack = 1218;
951     int numPositions = 1001; // 10nm/frame
952     float range = 10.0f; // total range 25.0 um -> 35.0 um
953     float umPerImg = range/numImgInStack;
954
955     QTrkComputedConfig cfg;
956     cfg.width=cfg.height = ROISize;
957     cfg.qi_angstep_factor = 1;
958     cfg.qi_iterations = 6;
959     cfg.qi_angular_coverage = 0.7f;
960     cfg.qi_roi_coverage = 1;
961     cfg.qi_radial_coverage = 1.5f;
962     cfg.qi_minradius=0;
963     cfg.zlut_minradius=0;
964     cfg.zlut_angular_coverage = 0.7f;
965     cfg.zlut_roi_coverage = 1;
966     cfg.zlut_radial_coverage = 1.5f;
967     cfg.zlut_minradius = 0;
968     cfg.qi_minradius = 0;
969     cfg.com_bgcorrection = 0;
970     cfg.xcl_profileLength = ROISize*0.8f;
971     cfg.xcl_profileWidth = ROISize*0.2f;
972     cfg.xcl_iterations = 1;
973     cfg.Update();
974     cfg.WriteToFile();
975
976     int zplanes = 50;
977
978     QueuedCUDATracker* qtrk = new QueuedCUDATracker(cfg);
979     //qtrk->SetLocalizationMode(LT_NormalizeProfile | LT_QI);
980     qtrk->SetRadialZLUT(0, beads.size(), zplanes);
981     qtrk->BeginLUT(0);
982
983     int pxPerBead = ROISize*ROISize;
984     int memSizePerBead = pxPerBead*sizeof(float);
985     int startFrame = 400;
986     for(int plane = 0; plane < zplanes; plane++){
987         output->outputString(SPrintf("Frame %d/%d",plane+1,zplanes),true);
988         int frameNum = startFrame+(int)(numImgInStack-startFrame)*((float)plane/zplanes);
989         std::string file = SPrintf("%s\\img%05d.jpg", folder.c_str(),frameNum);
990
991         ImageData frame = ReadJPEGFile(file.c_str());
992
993         float* data = new float[beads.size()*pxPerBead];
994
995         for(uint ii = 0; ii < beads.size(); ii++){
996             vector2f pos;
997             pos.x = beads.at(ii).x - ROISize/2;
998             pos.y = beads.at(ii).y - ROISize/2;
999             ImageData crop = CropImage(frame,pos.x,pos.y,ROISize,ROISize);
1000             //output->outputImage(crop,SPrintf("%d-%05d",ii,plane));
1001             memcpy(data+ii*pxPerBead,crop.data,memSizePerBead);
1002             crop.free();
1003         }
1004
1005         /*
1006         // To verify seperate frame bead stack generation
1007         output->newFile(SPrintf("data-plane-%d",plane));
1008         output->outputArray(data,beads.size()*pxPerBead);
1009
1010         ImageData allBeads = ImageData(data,ROISize,ROISize*beads.size());
1011         output->outputImage(allBeads,SPrintf("allBeads-%05d",frameNum));//*/
1012 }
```

```

1013     qtrk->BuildLUT(data, sizeof(float)*ROISize, QTrkFloat, plane);
1014
1015     frame.free();
1016     delete[] data;
1017 }
1018
1019 qtrk->FinalizeLUT();
1020 float* luts = new float[beads.size()*(zplanes*cfg.zlut_radialsteps)];
1021 qtrk->GetRadialZLUT(luts);
1022
1023 for(int ii = 0; ii < beads.size(); ii++){
1024     ImageData lut = ImageData::alloc(cfg.
1025     zlut_radialsteps, zplanes);
1026     memcpy(lut.data, &luts[ii*cfg.zlut_radialsteps*zplanes], cfg.
1027     zlut_radialsteps*zplanes*sizeof(float));
1028     //memcpy(lut.data,qtrk->GetZLUTByIndex(ii),cfg.zlut_radialsteps*zplanes*sizeof(float));
1029     //output->outputImage(lut,SPrintf("lut%03d,%d",beads.at(ii).x,beads.at(ii).y));
1030     output->outputImage(lut, SPrintf("lut%03d",ii));
1031     lut.free();
1032 }
1033 qtrk->Flush();
1034 delete qtrk;
1035 }
```

11.42.1.5 template<typename T> void check_arg (const std::vector< std::string > & args, const char * name, T * param)

```

744 {
745     for (uint i=0;i<args.size();i++) {
746         if (args[i] == name) {
747             *param = (T)atof(args[i+1].c_str());
748             return;
749         }
750     }
751 }
```

11.42.1.6 void check_strarg (const std::vector< std::string > & args, const char * name, std::string * param)

```

754 {
755     for (uint i=0;i<args.size();i++) {
756         if (args[i] == name) {
757             *param = args[i+1];
758             return;
759         }
760     }
761 }
```

11.42.1.7 int CmdLineRun (int argc, char * argv[])

```

764 {
765     QTrkSettings cfg;
766     std::vector<std::string> args(argc-1);
767     for (int i=0;i<argc-1;i++)
768         args[i]=argv[i+1];
769
770     check_arg(args, "roi", &cfg.width);
771     cfg.height=cfg.width;
772
773     int count=100;
774     check_arg(args, "count", &count);
775
776     std::string outfile, fixlutfile, inputposfile, bmlutfile, rescaledlutfile;
777     std::string radialWeightsFile;
778     check_strarg(args, "output", &outfile);
779     check_strarg(args, "fixlut", &fixlutfile);
780     check_strarg(args, "bmlut", &bmlutfile);
781     check_strarg(args, "inputpos", &inputposfile);
782     check_strarg(args, "regenlut", &rescaledlutfile);
783     check_strarg(args, "radweights", &radialWeightsFile);
784
785     std::string crloutput;
```

```

786     check_strarg(args, "crlb", &crlboutput);
787
788     std::vector< vector3f > inputPos;
789     if (!inputposfile.empty()) {
790         inputPos = ReadVector3CSV(inputposfile.c_str());
791         count = inputPos.size();
792     }
793
794     check_arg(args, "zlut_minradius", &cfg.zlut_minradius);
795     check_arg(args, "zlut_radial_coverage", &cfg.zlut_radial_coverage);
796     check_arg(args, "zlut_angular_coverage", &cfg.zlut_angular_coverage);
797     check_arg(args, "zlut_roi_coverage", &cfg.zlut_roi_coverage);
798
799     check_arg(args, "qi_iterations", &cfg.qi_iterations);
800     check_arg(args, "qi_minradius", &cfg.qi_minradius);
801     check_arg(args, "qi_radial_coverage", &cfg.qi_radial_coverage);
802     check_arg(args, "qi_angular_coverage", &cfg.qi_angular_coverage);
803     check_arg(args, "qi_roi_coverage", &cfg.qi_roi_coverage);
804     check_arg(args, "qi_angstep_factor", &cfg.qi_angstep_factor);
805     check_arg(args, "downsample", &cfg.downsample);
806
807     int zlutAlign=0;
808     check_arg(args, "zlutalign", &zlutAlign);
809
810     float pixelmax = 28 * 255;
811     check_arg(args, "pixelmax", &pixelmax);
812
813     std::string lutsmppfile;
814     check_strarg(args, "lutsmppfile", &lutsmppfile);
815
816     int cuda=1;
817     check_arg(args, "cuda", &cuda);
818     QueuedTracker* qtrk;
819
820     if (cuda) qtrk = new QueuedCUDATracker(cfg);
821     else qtrk = new QueuedCPUTracker(cfg);
822
823     ImageData lut;
824     BenchmarkLUT bmlut;
825
826     if (!fixlutfilename.empty())
827     {
828         lut = ReadJPEGFile(fixlutfilename.c_str());
829
830         if (!rescaledlutfilename.empty()) {
831             // rescaling allowed
832             ImageData newlut;
833             ResampleLUT(qtrk, &lut, lut.h, &newlut, rescaledlutfilename.c_str());
834             lut.free();
835             lut=newlut;
836         }
837         else if (lut.w != qtrk->cfg.zlut_radialsteps) {
838             lut.free();
839             dbgprintf("Invalid LUT size (%d). Expecting %d radialsteps\n", lut.
840 w, qtrk->cfg.zlut_radialsteps);
841             delete qtrk;
842             return -1;
843         }
844         qtrk->SetRadialZLUT(lut.data,1,lut.h);
845     }
846     else
847     {
848         if (bmlutfilename.empty()) {
849             delete qtrk;
850             dbgprintf("No lut file\n");
851             return -1;
852         }
853
854         bmlut.Load(bmlutfilename.c_str());
855         lut = ImageData::alloc(qtrk->cfg.zlut_radialsteps, bmlut.
856 lut_h);
857         bmlut.GenerateLUT(&lut);
858
859         if (!rescaledlutfilename.empty())
860             WriteJPEGFile(rescaledlutfilename.c_str(), lut);
861
862         qtrk->SetRadialZLUT(lut.data,1,lut.h);
863     }
864
865     if (inputPos.empty()) {
866         inputPos.resize(count);
867         for (int i=0;i<count;i++){
868             inputPos[i]=vector3f(cfg.width/2,cfg.height/2,lut.
869 h/2);
870         }
871     }

```

```

870     if (!radialWeightsFile.empty())
871     {
872         auto rwd = ReadCSV(radialWeightsFile.c_str());
873         std::vector<float> rw(rwd.size());
874         if (rw.size() == qtrk->cfg.zlut_radialsteps)
875             qtrk->SetRadialWeights(&rw[0]);
876         else
877             {
878                 dbprintf("Invalid # radial weights");
879                 delete qtrk;
880             }
881     }
882
883     std::vector<ImageData> imgs (inputPos.size());
884
885     std::vector<vector3f> crlb(inputPos.size());
886
887     for (uint i=0;i<inputPos.size();i++) {
888         imgs[i]=ImageData::alloc(cfg.width, cfg.height);
889         //vector3f pos = centerpos + range*vector3f(rand_uniform<float>()-0.5f, rand_uniform<float>()-0.5f,
890         rand_uniform<float>(-0.5f)*2;
891
892         auto p = inputPos[i];
893         if (!bmlut.lut_w) {
894             GenerateImageFromLUT(&imgs[i], &lut, qtrk->cfg.
895             zlut_minradius, qtrk->cfg.zlut_maxradius, p, false);
896             if (!crlboutput.empty()) {
897                 SampleFisherMatrix sfm(pixelmax);
898                 crlb[i]=sfm.Compute(p, vector3f(1,1,1)*0.001f, lut, qtrk->
899                 cfg.width,qtrk->cfg.height, qtrk->cfg.zlut_minradius, qtrk->
900                 cfg.zlut_maxradius).Inverse().diag();
901             }
902         } else
903             bmlut.GenerateSample(&imgs[i], p, qtrk->cfg.
904             zlut_minradius, qtrk->cfg.zlut_maxradius);
905         imgs[i].normalize();
906         if (pixelmax > 0) ApplyPoissonNoise(imgs[i], pixelmax, 255);
907         if(i==0 && !lutsmpfile.empty()) WriteJPEGFile(lutsmpfile.c_str(), imgs[i]);
908     }
909
910     int locMode = LT_LocalizeZ | LT_NormalizeProfile |
911     LT_LocalizeZWeighted;
912     if (qtrk->cfg.qi_iterations > 0)
913         locMode |= LT_QI;
914     if (zlutAlign)
915         locMode |= LT_ZLUTAlign;
916
917     qtrk->SetLocalizationMode((LocMode_t)locMode);
918     double tstart=GetPreciseTime();
919
920     for (uint i=0;i<inputPos.size();i++)
921     {
922         LocalizationJob job(i, 0, 0, 0);
923         qtrk->ScheduleImageData(&imgs[i], &job);
924     }
925
926     WaitForFinish(qtrk, inputPos.size());
927     double tend = GetPreciseTime();
928
929     std::vector<vector3f> results(inputPos.size());
930     for (uint i=0;i<inputPos.size();i++) {
931         LocalizationResult r;
932         qtrk->FetchResults(&r,1);
933         results[r.job.frame]=r.pos;
934     }
935     vector3f meanErr, stdevErr;
936     MeanStDevError(inputPos, results, meanErr, stdevErr);
937     dbprintf("Mean err X=%f,Z=%f. St deviation: X=%f,Z=%f\n", meanErr.
938     x,meanErr.y,stdevErr.x,stdevErr.z);
939
940     if (!crlboutput.empty())
941         WriteTrace(crlboutput, &crlb[0], crlb.size());
942     WriteTrace(outputfile, &results[0], inputPos.size());
943
944     if (lut.data) lut.free();
945     delete qtrk;
946
947     return 0;
948 }

```

11.42.1.8 void CompareAccuracy (const char * lutfile)

460 {

```

461     QTrkSettings cfg;
462     cfg.width=150;
463     cfg.height=150;
464     cfg.numThreads=1;
465
466     auto cpu = RunTracker<QueuedCPUTracker> (lutfile, &cfg, false, "cpu", LT_QI);
467     auto gpu = RunTracker<QueuedCUDATracker>(lutfile, &cfg, false, "gpu", LT_QI);
468 //    auto cpugc = RunTracker<QueuedCPUTracker>(lutfile, &cfg, true, "cpugc");
469 //    auto gpugc = RunTracker<QueuedCUDATracker>(lutfile, &cfg, true, "gpugc");
470
471     for (int i=0;i<std::min((int)cpu.output.size(),20);i++) {
472         dbgprintf("CPU-GPU: %f, %f\n", cpu.output[i].x-gpu.output[i].x,cpu.output[i].y-gpu.output[i].y);
473     }
474
475 /*    dbgprintf("CPU\tGPU\tCPU(gc)\tGPU(gc)\n");
476     dbgprintf("St Dev. : CPU: %.2f\tGPU: %.2f\tCPU(gc).%.2f\tGPU(gc).%.2f\n", StDev(cpu).x, StDev(gpu).x,
477     StDev(cpugc).x, StDev(gpugc).x);
478     dbgprintf("Mean err: CPU: %.2f\tGPU: %.2f\tCPU(gc).%.2f\tGPU(gc).%.2f\n", Mean(cpu).x, Mean(gpu).x,
479     Mean(cpugc).x, Mean(gpugc).x);
480 */
481 }
```

11.42.1.9 __device__ float compute (int *idx*, float * *buf*, int *s*)

```

65 {
66     // some random calcs to make the kernel unempty
67     float k=0.0f;
68     for (int x=0;x<s;x++ ){
69         k+=cosf(x*0.1f*idx);
70         buf[x]=k;
71     }
72     for (int x=0;x<s/2;x++){
73         buf[x]=buf[x]*buf[x];
74     }
75     float sum=0.0f;
76     for (int x=s-1;x>=1;x--) {
77         sum += buf[x-1]/(fabsf(buf[x])+0.1f);
78     }
79     return sum;
80 }
```

11.42.1.10 __global__ void emptyKernel ()

```
313 { }
```

11.42.1.11 std::string getPath (const char * *file*)

```

40 {
41     std::string s = file;
42     int pos = s.length()-1;
43     while (pos>0 && s[pos]!='\\' && s[pos]!='/' )
44         pos--;
45
46     return s.substr(0, pos);
47 }
```

11.42.1.12 void listDevices ()

```

256 {
257     cudaDeviceProp prop;
258     int dc;
259     cudaGetDeviceCount(&dc);
260     for (int k=0;k<dc;k++) {
261         cudaGetDeviceProperties(&prop, k);
262         dbgprintf("Device[%d] = %s\n", k, prop.name);
263         dbgprintf("\tMax texture width: %d\n", prop.maxTexture2D[0]);
264     }
265
266 }
```

11.42.1.13 int main (int argc, char * argv[])

```

1037 {
1038     //listDevices();
1039
1040     if (argc > 1)
1041     {
1042         return CmdLineRun(argc, argv);
1043     }
1044
1045     try {
1046         // outputter output(Files+Images);
1047         // BuildZLUT("C:\\TestImages\\TestMovie150507_2\\images\\jpg\\zstack\\", &output);
1048         BasicQTrkTest();
1049         // BasicQTrkTest_RM();
1050
1051         // TestBenchmarkLUT();
1052         // testLinearArray();
1053         // TestTextureFetch();
1054         // TestGauss2D(true);
1055         // MultipleLUTTest();
1056
1057         // TestSurfaceReadWrite();
1058         // TestImage4D();
1059         // TestImage4DMemory();
1060         // TestImageLUT("../cputrack-test/lut000.jpg");
1061         // TestRadialLUTGradientMethod();
1062
1063         // BenchmarkParams();
1064         // TestTextureFetch();
1065         // QICompare("../cputrack-test/lut000.jpg");
1066         // TestCMOSNoiseInfluence<QueuedCUDATracker>("../cputrack-test/lut000.jpg");
1067
1068         // CompareAccuracy("../cputrack-test/lut000.jpg");
1069         // QTrkCompareTest();
1070         /*
1071             ProfileSpeedVsROI(LT_OnlyCOM, "speeds-com.txt", false, 0);
1072             ProfileSpeedVsROI(LT_OnlyCOM, "speeds-com-z.txt", true, 0);
1073             ProfileSpeedVsROI(LT_XCor1D, "speeds-xcor.txt", true, 0);
1074             for (int qi_it=1;qi_it<4;qi_it++) {
1075                 ProfileSpeedVsROI(LT_QI, SPrintf("speeds-qi-%d-iterations.txt",qi_it).c_str(), true, qi_it);
1076             }*/
1077
1078         /* auto info = SpeedCompareTest(80, false);
1079             auto infogc = SpeedCompareTest(80, true);
1080             dbgprintf("[ginc=false] CPU: %f, GPU: %f\n", info.speed_cpu, info.speed_gpu);
1081             dbgprintf("[ginc=true] CPU: %f, GPU: %f\n", infogc.speed_cpu, infogc.speed_gpu);
1082         */
1083     } catch (const std::exception& e) {
1084         dbgprintf("Exception: %s\n", e.what());
1085     }
1086     system("pause");
1087     return 0;
1088 }
1089 }
```

11.42.1.14 __device__ float2 mul_conjugate (float2 a, float2 b) [inline]

```

50 {
51     float2 r;
52     r.x = a.x*b.x + a.y*b.y;
53     r.y = a.y*b.x - a.x*b.y;
54     return r;
55 }
```

11.42.1.15 int NearestPowerOfTwo (int v)

```

379 {
380     int r=1;
381     while (r < v)
382         r *= 2;
383     if ( fabsf(r-v) < fabsf(r/2-v) )
384         return r;
385     return r/2;
386 }
```

11.42.1.16 void ProfileSpeedVsROI (LocalizeModeEnum *locMode*, const char * *outputcsv*, bool *haveZLUT*, int *qi_iterations*)

```

445 {
446     std::vector<float> values;
447
448     for (int roi=20;roi<=180;roi+=10) { // same as BenchmarkROIAccuracy()
449         SpeedInfo info = SpeedCompareTest(roi, locMode, haveZLUT, qi_iterations);
450         values.push_back(roi);
451         values.push_back(info.speed_cpu);
452         values.push_back(info.speed_gpu);
453     }
454
455     const char *labels[] = { "ROI", "CPU", "CUDA" };
456     WriteImageAsCSV(outputcsv, &values[0], 3, values.size()/3, labels);
457 }
```

11.42.1.17 void QICompare (const char * *lutfile*)

```

675 {
676     QTrkSettings cfg;
677     cfg.qi_iterations=1;
678     cfg.width = 150;
679     cfg.height = 150;
680     cfg.numThreads=1;
681     QueuedCUDATracker gpu(cfg, 1);
682     QueuedCPUTracker cpu(cfg);
683
684     ImageData lut=ReadJPEGfile(lutfile);
685     ImageData img=ImageData::alloc(cfg.width,cfg.
686     height);
687
688     srand(0);
689     const int N=1;
690     gpu.SetLocalizationMode(LT_QI);
691     cpu.SetLocalizationMode(LT_QI);
692     for (int i=0;i<N;i++) {
693         LocalizationJob job(i, 0, 0, 0);
694         vector3f pos(cfg.width/2,cfg.height/2, lut.h/2);
695         pos.x += rand_uniform<float>();
696         pos.y += rand_uniform<float>();
697         GenerateImageFromLUT(&img, &lut, 1, cfg.width/2, pos);
698         gpu.ScheduleLocalization( (uchar*)img.data, sizeof(float)*img.w,
699         QTrkFloat, &job);
700         cpu.ScheduleLocalization( (uchar*)img.data, sizeof(float)*img.w,
701         QTrkFloat, &job);
702     }
703     gpu.Flush();
704     cpu.Flush();
705     while(cpu.GetResultCount() != N || gpu.GetResultCount() != N );
706
707     auto rcpu = FetchResults(&cpu), rgpu = FetchResults(&gpu);
708     for (int i=0;i<N;i++) {
709         vector3f d=rcpu[i]-rgpu[i];
710         dbgprintf("[%d]: CPU: x=%f, y=%f. GPU: x=%f, y=%f.\tGPU-CPU: x:%f, y:%f\n", i, rcpu[i].x,
711         rcpu[i].y, rgpu[i].x, rgpu[i].y, d.x,d.y);
712
713     // Profiles
714     for(uint i=0;i<cmp_cpu_qi_prof.size();i++) {
715         dbgprintf("QIPROF[%d]. CPU=%f, GPU=%f, Diff: %f\n", i,
716         cmp_cpu_qi_prof[i], cmp_gpu_qi_prof[i],
717         cmp_gpu_qi_prof[i]-cmp_cpu_qi_prof[i]);
718     }
719     // FFT out
720     for(uint i=0;i<cmp_cpu_qi_fft_out.size();i++) {
721         dbgprintf("fft-out[%d]. CPU=%f, GPU=%f, Diff: %f\n", i,
722         cmp_cpu_qi_fft_out[i].real(), cmp_gpu_qi_fft_out[i].real(),
723         cmp_gpu_qi_fft_out[i].real()-cmp_cpu_qi_fft_out[i].real());
724     }
725     img.free();
726     lut.free();
727 }
```

11.42.1.18 void QTrkCompareTest()

```

120 {
121     QTrkSettings cfg;
122     cfg.width = cfg.height = 40;
123     cfg.qi_iterations = 1;
124     cfg.xcl_iterations = 2;
125     cfg.xcl_profileLength = 64;
126     cfg.numThreads = -1;
127     cfg.com_bgcorrection = 0.0f;
128     bool haveZLUT = false;
129 #ifdef _DEBUG
130     cfg.numThreads = 2;
131     cfg.qi_iterations=1;
132     int total= 10;
133     int batchSize = 2;
134     haveZLUT=false;
135 #else
136     cfg.numThreads = 4;
137     int total = 10000;
138     int batchSize = 512;
139 #endif
140
141     QueuedCUDATracker qtrk(cfg, batchSize);
142     QueuedCPUTracker qtrkcpu(cfg);
143     ImageData img = ImageData::alloc(cfg.width,cfg.
144                                     height);
145     bool cpucmp = true;
146
147     qtrk.EnableTextureCache(true);
148
149     srand(1);
150
151     // Generate ZLUT
152     int zplanes=100;
153     float zmin=0.5,zmax=3;
154     qtrk.SetRadialZLUT(0, 1, zplanes);
155     if (cpucmp) qtrkcpu.SetRadialZLUT(0, 1, zplanes);
156     if (haveZLUT) {
157         for (int x=0;x<zplanes;x++) {
158             vector2f center ( cfg.width/2, cfg.height/2 );
159             float s = zmin + (zmax-zmin) * x/(float)(zplanes-1);
160             GenerateTestImage(img, center.x, center.y, s, 0.0f);
161             WriteJPEGFile("qtrkzlutimg.jpg", img);
162
163             qtrk.BuildLUT(img.data,img.pitch(),QTrkFloat, 0, (
164             vector2f*)(0));
165             if (cpucmp)
166                 qtrkcpu.BuildLUT(img.data,img.pitch(),QTrkFloat, 0);
167
168             qtrk.FinalizeLUT();
169             if (cpucmp) qtrkcpu.FinalizeLUT();
170             // wait to finish ZLUT
171             while(true) {
172                 int rc = qtrk.GetResultCount();
173                 if (rc == zplanes) break;
174                 Sleep(100);
175                 dbgprintf(".");
176             }
177         }
178     }
179     float* zlut = new float[qtrk.cfg.zlut_radialsteps*zplanes];
180     qtrk.GetRadialZLUT(zlut);
181     if (cpucmp) {
182         float* zlutcpu = new float[qtrkcpu.cfg.zlut_radialsteps*zplanes];
183         qtrkcpu.GetRadialZLUT(zlutcpu);
184
185         WriteImageAsCSV("zlut-cpu.txt", zlutcpu, qtrkcpu.cfg.zlut_radialsteps, zplanes);
186         WriteImageAsCSV("zlut-gpu.txt", zlut, qtrkcpu.cfg.zlut_radialsteps, zplanes);
187         delete[] zlutcpu;
188     }
189     qtrk.ClearResults();
190     if (cpucmp) qtrkcpu.ClearResults();
191     FloatToJPEGFile ("qtrkzlutcuda.jpg", zlut, qtrk.cfg.zlut_radialsteps, zplanes);
192     delete[] zlut;
193
194     // Schedule images to localize on
195     dbgprintf("Benchmarking...\n", total);
196     GenerateTestImage(img, cfg.width/2, cfg.height/2, (zmin+zmax)/2, 0);
197     double tstart = GetPreciseTime();
198     int rc = 0, displayrc=0;
199     LocMode_t flags = (LocMode_t)(LT_NormalizeProfile |
200     LT_QI| (haveZLUT ? LT_LocalizeZ : 0 ) );
201     qtrk.SetLocalizationMode(flags);

```

```

201     qtrkcpu.SetLocalizationMode(flags);
202     for (int n=0;n<ntotal;n++) {
203         LocalizationJob jobInfo;
204         jobInfo.frame = n;
205         jobInfo.zlutIndex = 0;
206         qtrk.ScheduleLocalization((uchar*)img.data, cfg.width*sizeof(float),
207 QTrkFloat,&jobInfo);
207         if (cpucmp) qtrkcpu.ScheduleLocalization((uchar*)img.data, cfg.
208 width*sizeof(float), QTrkFloat, &jobInfo);
208         if (n % 10 == 0) {
209             rc = qtrk.GetResultCount();
210             while (displayrc<rc) {
211                 if( displayrc%(total/10)==0) dbgprintf("Done: %d / %d\n", displayrc, total);
212                 displayrc++;
213             }
214         }
215     }
216     if (cpucmp) qtrkcpu.Flush();
217     WaitForFinish(&qtrk, total);
218
219     // Measure speed
220     double tend = GetPreciseTime();
221
222     if (cpucmp) {
223         dbgprintf("waiting for cpu results..\n");
224         while (total != qtrkcpu.GetResultCount())
225             Sleep(10);
226     }
227
228
229     img.free();
230
231     const int NumResults = 20;
232     LocalizationResult results[NumResults], resultscpu[NumResults];
233     int rcount = std::min(NumResults,total);
234     for (int i=0;i<rcount;i++) {
235         qtrk.FetchResults(&results[i], 1);
236         if (cpucmp) qtrkcpu.FetchResults(&resultscpu[i], 1);
237     }
238
239     // if you wonder about this syntax, google C++ lambda functions
240     std::sort(results, results+rcount, [](LocalizationResult a,
241 LocalizationResult b) -> bool { return a.job.frame > b.
242 job.frame; });
243     if(cpucmp) std::sort(resultscpu, resultscpu+rcount, [](LocalizationResult a,
244 LocalizationResult b) -> bool { return a.job.frame > b.
245 job.frame; });
246     for (int i=0;i<rcount;i++) {
247         LocalizationResult& r = results[i];
248         dbgprintf("gpu [%d] x: %f, y: %f. z: %g, COM: %f, %f\n", i,r.
249 pos.x, r.pos.y, r.pos.z, r.firstGuess.x, r.firstGuess.
250 y);
251
252         if (cpucmp) {
253             r = resultscpu[i];
254             dbgprintf("cpu [%d] x: %f, y: %f. z: %g, COM: %f, %f\n", i,r.
255 pos.x, r.pos.y, r.pos.z, r.firstGuess.x, r.firstGuess.
256 y);
257         }
258     }
259     dbgprintf("Localization Speed: %d (img/s)\n", (int)( total/(tend-tstart) ));
260 }
```

11.42.1.19 void ShowCUDAError()

```

57
58     cudaError_t err = cudaGetLastError();
59     dbgprintf("Cuda error: %s\n", cudaGetStringError(err));
60 }
```

11.42.1.20 __global__ void SimpleKernel(int N, float * a)

```

268
269     int idx = blockIdx.x * blockDim.x + threadIdx.x;
270     if (idx < N) {
271         for (int x=0;x<1000;x++)
272             a[idx] = asin(a[idx]+x);
273     }
274 }
```

11.42.1.21 int SmallestPowerOfTwo (int minval)

```

389 {
390     int r=1;
391     while (r < minval)
392         r *= 2;
393     return r;
394 }

```

11.42.1.22 SpeedInfo SpeedCompareTest (int w, LocalizeModeEnum locMode, bool haveZLUT, int qi_iterations = 5)

```

402 {
403     int cudaBatchSize = 1024;
404     int count = 60000;
405
406 #ifdef _DEBUG
407     count = 100;
408     cudaBatchSize = 32;
409 #endif
410     LocMode_t locType = (LocMode_t)( locMode |
411                                     LT_NormalizeProfile );
412
413     QTrkComputedConfig cfg;
414     cfg.width = cfg.height = w;
415     cfg.qi_iterations = qi_iterations;
416     cfg.qi_radial_coverage = 1.5f;
417     cfg.qi_angstep_factor = 1.5f;
418     cfg.qi_angular_coverage = 0.7f;
419     cfg.zlut_radial_coverage = 2.0f;
420     //std::vector<int> devices(1); devices[0]=1;
421     //SetCUDADevices(devices);
422     cfg.cuda_device = QTrkCUDA_UseAll;
423     cfg.numThreads = -1;
424     cfg.com_bgcorrection = 0.0f;
425     cfg.Update();
426     dbgprintf("Width: %d, QI radius: %f, radialsteps: %d\n", w, cfg.
427     qimaxradius, cfg.qi_radialsteps);
428
429     SpeedInfo info;
430     QueuedCPUTracker *cputrk = new QueuedCPUTracker(cfg);
431     info.speed_cpu = SpeedTest(cfg, cputrk, count, haveZLUT, locType, &info.
432     sched_cpu, false);
433     delete cputrk;
434
435     QueuedCUDATracker *cudatrk = new QueuedCUDATracker(cfg, cudaBatchSize
436     );
437     info.speed_gpu = SpeedTest(cfg, cudatrk, count, haveZLUT, locType, &info.
438     sched_gpu, false);
439     //info.speed_gpu = SpeedTest(cfg, cudatrk, count, haveZLUT, locType, &info.sched_gpu);
440     std::string report = cudatrk->GetProfileReport();
441     delete cudatrk;
442
443     dbgprintf("CPU tracking speed: %d img/s\n", (int)info.speed_cpu);
444     dbgprintf("GPU tracking speed: %d img/s\n", (int)info.speed_gpu);
445
446     return info;
447 }

```

11.42.1.23 float SpeedTest (const QTrkSettings & cfg, QueuedTracker * qtrk, int count, bool haveZLUT, LocMode_t locType, float * scheduleTime, bool gainCorrection = false)

qtrk->ScheduleLocalization((uchar*)image, cfg.width*sizeof(float), QTrkFloat, flags, n, 0, 0, 0, 0);

```

316 {
317     ImageData img=ImageData::alloc(cfg.width,cfg.
318     height);
319     srand(1);
320     // Generate ZLUT
321     int zplanes=100;
322     float zmin=0.5,zmax=3;
323     qtrk->SetRadialZLUT(0, 1, zplanes);
324     if (gainCorrection) EnableGainCorrection(qtrk);

```

```

325     if (haveZLUT) {
326         for (int x=0;x<zplanes;x++) {
327             vector2f center( cfg.width/2, cfg.height/2 );
328             float s = zmin + (zmax-zmin) * x/(float)(zplanes-1);
329             GenerateTestImage(img, center.x, center.y, s, 0.0f);
330             qtrk->BuildLUT(img.data,img.pitch(),QTrkFloat, 0);
331         }
332         qtrk->FinalizeLUT();
333     }
334     qtrk->ClearResults();
335
336     // Schedule images to localize on
337     dbgprintf("Benchmarking...\n", count);
338     GenerateTestImage(img, cfg.width/2, cfg.height/2, (zmin+zmax)/2, 0);
339     double tstart = GetPreciseTime();
340     int rc = 0, displayrc=0;
341     double maxScheduleTime = 0.0f;
342     double sumScheduleTime2 = 0.0f;
343     double sumScheduleTime = 0.0f;
344     qtrk->SetLocalizationMode(locType| (haveZLUT ?
345     LT_LocalizeZ : 0));
346     for (int n=0;n<count;n++) {
347         double t0 = GetPreciseTime();
348         ROIPosition roiPos[]={{ {0,0} } };
349         LocalizationJob job(n, 0, 0,0);
350         qtrk->ScheduleFrame((uchar*)img.data, cfg.width*sizeof(float),cfg.
351 width,cfg.height, roiPos, 1, QTrkFloat, &job);
352         double dt = GetPreciseTime() - t0;
353         maxScheduleTime = std::max(maxScheduleTime, dt);
354         sumScheduleTime += dt;
355         sumScheduleTime2 += dt*dt;
356
357         if (n % 10 == 0) {
358             rc = qtrk->GetResultCount();
359             while (displayrc<rc) {
360                 if( displayrc%(count/10)==0) dbgprintf("Done: %d / %d\n", displayrc, count);
361                 displayrc++;
362             }
363         }
364     }
365     WaitForFinish(qtrk, count);
366
367     // Measure speed
368     double tend = GetPreciseTime();
369     img.free();
370
371     float mean = sumScheduleTime / count;
372     float stdev = sqrt(sumScheduleTime2 / count - mean * mean);
373     dbgprintf("Scheduletime: Avg=%f, Max=%f, Stdev=%f\n", mean*1000, maxScheduleTime*1000, stdev*1
000);
374     *scheduleTime = mean;
375
376     return count/(tend-tstart);
377 }
```

11.42.1.24 void TestAsync()

```

277 {
278     int N =100000;
279     int nt = 32;
280
281     pinned_array<float> a(N);
282 //    cudaMallocHost(&a, sizeof(float)*N, 0);
283
284     device_vec<float> A(N);
285
286     cudaStream_t s0;
287     cudaEvent_t done;
288
289     cudaStreamCreate(&s0);
290     cudaEventCreate(&done,0);
291
292     for (int x=0;x<N;x++)
293         a[x] = cos(x*0.01f);
294
295     for (int x=0;x<1;x++) {
296         { MeasureTime mt("a->A"); A.copyToDevice(a.data(), N, true); }
297         { MeasureTime mt("func(A)");
298             SimpleKernel<<<dim3( (N+nt-1)/nt ), dim3(nt)>>>(N, A.data);
299         }
300         { MeasureTime mt("A->a"); A.copyToHost(a.data(), true); }
301     }
```

```

302     cudaEventRecord(done);
303
304     {
305         MeasureTime("sync..."); while (cudaEventQuery(done) != cudaSuccess);
306     }
307
308     cudaStreamDestroy(s0);
309     cudaEventDestroy(done);
310 }
```

11.42.1.25 void TestBenchmarkLUT()

```

727 {
728     BenchmarkLUT bml("refbeadlut.jpg");
729
730     ImageData img=ImageData::alloc(120,120);
731
732     ImageData lut = ImageData::alloc(bml.lut_w, bml.lut_h);
733     bml.GenerateLUT(&lut);
734     WriteJPEGFile("refbeadlut-lutsmp.jpg", lut);
735     lut.free();
736
737     bml.GenerateSample(&img, vector3f(img.w/2,img.h/2,bml.lut_h/2), 0, img.
738 w/2-5);
739     WriteJPEGFile("refbeadlut-bmsmp.jpg", img);
740     img.free();
741 }
```

11.42.1.26 void TestGauss2D (bool *calib*)

```

649 {
650     int N=20, R=1000;
651 #ifdef _DEBUG
652     R=1;
653 #endif
654     std::vector<vector3f> rcpu = Gauss2DTest<QueuedCPUTracker>(N, R, calib);
655     std::vector<vector3f> rgpu = Gauss2DTest<QueuedCUDATracker>(N, R, calib);
656
657     for (int i=0;i<std::min(20,N);i++) {
658         dbgprintf("[%d] CPU: X:%.5f, Y:%.5f\tGPU: X:%.5f, Y:%.5f \tDiff: X:%.5f, Y:%.5f\n",
659                     i, rcpu[i].x, rcpu[i].y, rgpu[i].x, rgpu[i].y, rcpu[i].x-rgpu[i].x, rcpu[i].y-rgpu[i].y);
660     }
661 }
```

11.42.1.27 void TestRadialLUTGradientMethod()

```

664 {
665
666 }
```

11.42.1.28 void TestSharedMem()

```

97 {
98     int n=100, s=200;
99     dim3 nthreads(32), nblocks( (n+nthreads.x-1)/nthreads.x);
100    device_vec<float> buf(n*s);
101    device_vec<float> result_s(n), result_g(n);
102
103    double t0 = GetPreciseTime();
104    testWithGlobal<<<nblocks,nthreads>>>(n,s,result_g.data,buf.data);
105    cudaDeviceSynchronize();
106    double t1 = GetPreciseTime();
107    testWithShared <<<nblocks,nthreads,s*sizeof(float)*nthreads.x>>>(n,s,result_s.data);
108    cudaDeviceSynchronize();
109    double t2 = GetPreciseTime();
110
111    std::vector<float> rs = result_s, rg = result_g;
112    for (int x=0;x<n;x++) {
113        dbgprintf("result_s[%d]=%f. result_g[%d]=%f\n", x,rs[x], x,rg[x]);
114    }
115
116    dbgprintf("Speed of shared comp: %f, speed of global comp: %f\n", n/(t2-t1), n/(t1-t0));
117 }
```

11.42.1.29 `__global__ void testWithGlobal(int n, int s, float * result, float * buf)`

```

82
83     int idx = threadIdx.x + blockIdx.x * blockDim.x;
84     if (idx < n) {
85         result [idx] = compute(idx, &buf [idx * s], s);
86     }
87 }
```

11.42.1.30 `__global__ void testWithShared(int n, int s, float * result)`

```

89
90     int idx = threadIdx.x + blockIdx.x * blockDim.x;
91     if (idx < n) {
92         result [idx] = compute(idx, &cudaSharedMem[threadIdx.x * s], s);
93     }
94 }
```

11.42.2 Variable Documentation

11.42.2.1 `std::vector< std::complex<float> > cmp_cpu_qi_fft_out`

11.42.2.2 `std::vector< float > cmp_cpu_qi_prof`

11.42.2.3 `std::vector< std::complex<float> > cmp_gpu_qi_fft_out`

11.42.2.4 `std::vector< float > cmp_gpu_qi_prof`

11.42.2.5 `__shared__ float cudaSharedMem[]`

11.43 cudatrack/cudalmageList.h File Reference

```
#include "gpu_utils.h"
```

Classes

- struct `cudalmageList< T >`
- struct `Image4DCudaArray< T >`
- struct `Image4DCudaArray< T >::KernelInst`
- class `Image4DMemory< T >`
- struct `Image4DMemory< T >::KernelParams`

11.44 cudatrack/gpu_utils.h File Reference

```
#include <cuda_runtime.h>
#include <vector>
#include <cstdarg>
#include "cufft.h"
#include "LsqQuadraticFit.h"
```

Classes

- class `device_vec< T >`
- struct `MeasureTime`
- class `pinned_array< T, flags >`

Macros

- `#define CUDA_SUPPORTED_FUNC __device__ __host__`
- `#define CUBOTH __device__ __host__`

Functions

- void `outputTotalGPUMemUse` (std::string info="")
- void `CheckCUDAError` (cufftResult_t err)
- void `CheckCUDAError` (cudaError_t err)
- void `CheckCUDAError` ()
- void `dbgCUDAErrorCheck` (cudaError_t e)
- void `DbgCopyResult` (`device_vec< float2 >` src, std::vector< std::complex< float > > &dst)
- void `DbgCopyResult` (`device_vec< float >` src, std::vector< float > &dst)
- void `DbgOutputVectorToFile` (std::string loc, `device_vec< float >` &src, bool append)

11.44.1 Macro Definition Documentation

11.44.1.1 `#define CUBOTH __device__ __host__`

11.44.1.2 `#define CUDA_SUPPORTED_FUNC __device__ __host__`

11.44.2 Function Documentation

11.44.2.1 `void CheckCUDAError(cufftResult_t err) [inline]`

```
36 {
37     if (err != CUFFT_SUCCESS) {
38         outputTotalGPUMemUse("CUFFT Error");
39         throw std::runtime_error(Sprintf("CUDA error: CUFFT failed (%d)\n",err));
40     }
41 }
```

11.44.2.2 `void CheckCUDAError(cudaError_t err) [inline]`

```
44 {
45     if (err != cudaSuccess) {
46         const char* errstr = cudaGetErrorString(err);
47         throw std::runtime_error(Sprintf("CUDA error: %s\n" ,errstr).c_str());
48     }
49 }
```

11.44.2.3 void CheckCUDAError() [inline]

```

52 {
53     cudaError_t err = cudaGetLastError();
54     if (err != cudaSuccess) {
55         const char* errstr = cudaGetStringError(err);
56         dbgprintf("CUDA error: %s\n", errstr);
57     }
58 }
```

11.44.2.4 void DbgCopyResult(device_vec<float2> src, std::vector<std::complex<float>> & dst) [inline]

```
266 {}
```

11.44.2.5 void DbgCopyResult(device_vec<float> src, std::vector<float> & dst) [inline]

```
267 {}
```

11.44.2.6 void dbgCUDAErrorCheck(cudaError_t e) [inline]

```
62 {}
```

11.44.2.7 void DbgOutputVectorToFile(std::string loc, device_vec<float> & src, bool append) [inline]

```
268 {}
```

11.44.2.8 void outputTotalGPUMemUse(std::string info = "") [inline]

```

18 {
19     // show total memory usage of GPU
20     size_t free_byte;
21     size_t total_byte;
22     cudaError_t cuda_status = cudaMemGetInfo( &free_byte, &total_byte );
23     if ( cudaSuccess != cuda_status ){
24         dbgprintf("Error: cudaMemGetInfo fails, %s \n", cudaGetStringError(cuda_status) );
25         exit(1);
26     }
27     double free_db = (double)free_byte;
28     double total_db = (double)total_byte;
29     double used_db = total_db - free_db;
30     dbgprintf("%sused = %2.2f MB, free = %2.2f MB, total = %2.2f MB\n",
31               info != "" ? (info+":").c_str() : "",
32               used_db/1024.0/1024.0, free_db/1024.0/1024.0, total_db/1024.0/1024.0);
33 }
```

11.45 cudatrack/ImageSampler.h File Reference

```
#include "cudaImageList.h"
```

Classes

- class `ImageSampler_MemCopy`
- class `ImageSampler_SimpleTextureRead`
- class `ImageSampler_InterpolatedTexture`

Typedefs

- typedef `ImageSampler_InterpolatedTexture ImageSampler_Tex`

Functions

- `texture< float, cudaTextureType2D, cudaReadModeElementType > image_sampler_texture_linear (0, cudaFilterModeLinear)`
- `texture< float, cudaTextureType2D, cudaReadModeElementType > image_sampler_texture_nearest (0, cudaFilterModePoint)`

11.45.1 Typedef Documentation

11.45.1.1 typedef `ImageSampler_InterpolatedTexture ImageSampler_Tex`

11.45.2 Function Documentation

11.45.2.1 `texture<float, cudaTextureType2D, cudaReadModeElementType> image_sampler_texture_linear (0 , cudaFilterModeLinear)`

11.45.2.2 `texture<float, cudaTextureType2D, cudaReadModeElementType> image_sampler_texture_nearest (0 , cudaFilterModePoint)`

11.46 cudatrack/Kernels.h File Reference

Functions

- template<typename T>
`static __device__ T interpolate (T a, T b, float x)`
- template<typename TIImageSampler>
`__device__ float2 BgCorrectedCOM (int idx, cudalImageListf images, float correctionFactor, float *pMean)`
- template<typename TIImageSampler>
`__global__ void BgCorrectedCOM (int count, cudalImageListf images, float3 *d_com, float bgCorrection← Factor, float *d_imgmeans)`
- `__global__ void ZLUT_ProfilesToZLUT (int njobs, cudalImageListf images, ZLUTParams params, float3 *positions, LocalizationParams *locParams, float *profiles)`
- template<typename TIImageSampler>
`__global__ void ZLUT_RadialProfileKernel (int njobs, cudalImageListf images, ZLUTParams params, float3 *positions, float *profiles, float *means)`
- `__global__ void ZLUT_ComputeZ (int njobs, ZLUTParams params, float3 *positions, float *compareScore← Buf)`
- `__global__ void ZLUT_ComputeProfileMatchScores (int njobs, ZLUTParams params, float *profiles, float *compareScoreBuf, LocalizationParams *locParams)`

- `__global__ void ZLUT_NormalizeProfiles (int njobs, ZLUTParams params, float *profiles)`
- `__global__ void ApplyOffsetGain (BaseKernelParams kp, cudalmageListf calib_gain, cudalmageListf calib_offset, float gainFactor, float offsetFactor)`
- template<typename TImageSampler >
`__global__ void G2MLE_Compute (BaseKernelParams kp, float sigma, int iterations, float3 *initial, float3 *positions, float *l_bg, float *l_0)`
- template<typename TImageSampler , typename TImageLUT >
`__global__ void ImageLUT_Sample (BaseKernelParams kp, float2 ilut_scale, float3 *positions, typename TImageLUT::KernelParams lut)`
- `__global__ void ForceCUDAKernelToLoad ()`

Variables

- surface< void, cudaSurfaceType2DLayered > `image_lut_surface`

11.46.1 Function Documentation

11.46.1.1 `__global__ void ApplyOffsetGain (BaseKernelParams kp, cudalmageListf calib_gain, cudalmageListf calib_offset, float gainFactor, float offsetFactor)`

```

163 {
164     int x = threadIdx.x + blockIdx.x * blockDim.x;
165     int y = threadIdx.y + blockIdx.y * blockDim.y;
166     int jobIdx = threadIdx.z + blockIdx.z * blockDim.z;
167
168     if (x < kp.images.w && y < kp.images.h && jobIdx < kp.njobs) {
169         int bead = kp.locParams[jobIdx].zlutIndex;
170
171         float value = kp.images.pixel(x,y,jobIdx);
172         float offset = calib_offset.isEmpty() ? 0 : calib_offset.pixel(x,y,bead);
173         float gain = calib_gain.isEmpty() ? 1 : calib_gain.pixel(x,y,bead);
174         kp.images.pixel(x,y,jobIdx) = (value + offset*offsetFactor) * gain*gainFactor;
175     }
176 }
```

11.46.1.2 template<typename TImageSampler > `__device__ float2 BgCorrectedCOM (int idx, cudalmageListf images, float correctionFactor, float * pMean)`

```

11 {
12     int imgsize = images.w*images.h;
13     float sum=0, sum2=0;
14     float momentX=0;
15     float momentY=0;
16
17     for (int y=0;y<images.h;y++)
18         for (int x=0;x<images.w;x++) {
19             float v = TImageSampler::Index(images, x, y, idx);
20             sum += v;
21             sum2 += v*v;
22         }
23
24     float invN = 1.0f/imgsize;
25     float mean = sum * invN;
26     float stdev = sqrtf(sum2 * invN - mean * mean);
27     sum = 0.0f;
28
29     for (int y=0;y<images.h;y++)
30         for(int x=0;x<images.w;x++)
31     {
32         float v = TImageSampler::Index(images, x,y,idx);
33         v = fabsf(v-mean)-correctionFactor*stdev;
34         if(v<0.0f) v=0.0f;
35         sum += v;
36         momentX += x*v;
37         momentY += y*v;
38     }
```

```

39     if (pMean)
40         *pMean = mean;
41
42     float2 com;
43     com.x = momentX / (float)sum;
44     com.y = momentY / (float)sum;
45
46     return com;
47 }

```

11.46.1.3 template<typename TImageSampler> __global__ void BgCorrectedCOM (int count, cudalimageListf images, float3 * d_com, float bgCorrectionFactor, float * d_imgmeans)

```

50
51     {
52         int idx = threadIdx.x + blockDim.x * blockIdx.x;
53         if (idx < count) {
54             float mean;
55             float2 com = BgCorrectedCOM<TImageSampler> (idx, images, bgCorrectionFactor, &mean);
56             d_com[idx] = make_float3(com.x, com.y, 0.0f);
57             d_imgmeans[idx] = mean;
58     }
59 }

```

11.46.1.4 __global__ void ForceCUDAKernelstoLoad ()

```

287 {
288 }

```

11.46.1.5 template<typename TImageSampler> __global__ void G2MLE_Compute (BaseKernelParams kp, float sigma, int iterations, float3 * initial, float3 * positions, float * l_bg, float * l_0)

```

182 {
183     int jobIdx = threadIdx.x + blockIdx.x * blockDim.x;
184
185     if (jobIdx >= kp.njobs)
186         return;
187
188     float2 pos = make_float2(initial[jobIdx].x, initial[jobIdx].y);
189     float mean = kp.imgmeans[jobIdx];
190     float I0 = mean*0.5f*kp.images.w*kp.images.h;
191     float bg = mean*0.5f;
192
193     const float _loSq2Sigma = 1.0f / (sqrtf(2) * sigma);
194     const float _loSq2PiSigma = (1.0f / (sqrtf(2*3.14159265359f))) / sigma;
195     const float _loSq2PiSigma3 = (1.0f / (sqrtf(2*3.14159265359f))) / (sigma*sigma*sigma);
196
197     for (int i=0;i<iterations;i++)
198     {
199         float dL_dx = 0.0;
200         float dL_dy = 0.0;
201         float dL_dI0 = 0.0;
202         float dL_dIbg = 0.0;
203         float dL2_dx = 0.0;
204         float dL2_dy = 0.0;
205         float dL2_dI0 = 0.0;
206         float dL2_dIbg = 0.0;
207
208         for (int y=0;y<kp.images.h;y++)
209         {
210             for (int x=0;x<kp.images.w;x++)
211             {
212                 float Xexp0 = (x-pos.x + .5f) * _loSq2Sigma;
213                 float Yexp0 = (y-pos.y + .5f) * _loSq2Sigma;
214
215                 float Xexp1 = (x-pos.x - .5f) * _loSq2Sigma;
216                 float Yexp1 = (y-pos.y - .5f) * _loSq2Sigma;
217
218                 float DeltaX = 0.5f * erff(Xexp0) - 0.5f * erff(Xexp1);
219                 float DeltaY = 0.5f * erff(Yexp0) - 0.5f * erff(Yexp1);

```

```

220         float mu = bg + I0 * DeltaX * DeltaY;
221
222         float dmu_dx = I0*_loSq2PiSigma * ( expf(-Xexp1*Xexp1) - expf(-Xexp0*Xexp0) ) * DeltaY;
223
224         float dmu_dy = I0*_loSq2PiSigma * ( expf(-Yexp1*Yexp1) - expf(-Yexp0*Yexp0) ) * DeltaX;
225         float dmu_dI0 = DeltaX*DeltaY;
226         float dmu_dIbg = 1;
227
228         float smp = TImageSampler::Index(kp.images, x, y, jobIdx);
229         float f = smp / mu - 1;
230         dL_dx += dmu_dx * f;
231         dL_dy += dmu_dy * f;
232         dL_dI0 += dmu_dI0 * f;
233         dL_dIbg += dmu_dIbg * f;
234
235         float d2mu_dx = I0*_loSq2PiSigma3 * ( (x - pos.x - .5f) * expf (-Xexp1*Xexp1) - (x - pos.x
236 + .5) * expf(-Xexp0*Xexp0) ) * DeltaX;
237         float d2mu_dy = I0*_loSq2PiSigma3 * ( (y - pos.y - .5f) * expf (-Yexp1*Yexp1) - (y - pos.y
238 + .5) * expf(-Yexp0*Yexp0) ) * DeltaY;
239         dL2_dx += d2mu_dx * f - dmu_dx*dmu_dx * smp / (mu*mu);
240         dL2_dy += d2mu_dy * f - dmu_dy*dmu_dy * smp / (mu*mu);
241         dL2_dI0 += -dmu_dI0*dmu_dI0 * smp / (mu*mu);
242         dL2_dIbg += -smp / (mu*mu);
243     }
244
245     pos.x -= dL_dx / dL2_dx;
246     pos.y -= dL_dy / dL2_dy;
247     I0 -= dL_dI0 / dL2_dI0;
248     bg -= dL_dIbg / dL2_dIbg;
249
250
251     positions[jobIdx].x = pos.x;
252     positions[jobIdx].y = pos.y;
253     if (I_bg) I_bg[jobIdx] = bg;
254     if (I_0) I_0[jobIdx] = I0;
255 }
```

11.46.1.6 template<typename TImageSampler , typename TImageLUT > __global__ void ImageLUT_Sample (BaseKernelParams kp, float2 ilut_scale, float3 * positions, typename TImageLUT::KernelParams lut)

```

262 {
263     // add sampled image data to
264     int x = threadIdx.x + blockIdx.x * blockDim.x;
265     int y = threadIdx.y + blockIdx.y * blockDim.y;
266     int id = threadIdx.z + blockIdx.z * blockDim.z;
267     if (x < lut.imgw && y < lut.imgh && id < kp.njobs) {
268
269         float invMean = 1.0f / kp.imgmeans[id];
270
271         float startx = positions[id].x - lut.imgw/2*ilut_scale.x;
272         float starty = positions[id].y - lut.imgh/2*ilut_scale.y;
273         int2 imgpos = lut.GetImagePos(kp.locParams[id].zlutPlane, kp.
274             locParams[id].zlutIndex);
275
276         float px = startx + x*ilut_scale.x;
277         float py = starty + y*ilut_scale.y;
278
279         bool outside=false;
280         float v = TImageSampler::Interpolated(kp.images, px, py, id, outside);
281
282         float org = TImageLUT::read(lut, x, y, imgpos);
283         TImageLUT::write(org+v*invMean, lut, x, y, imgpos);
284     }
285 }
```

11.46.1.7 template<typename T > static __device__ T interpolate (T a, T b, float x) [static]

```
7 { return a + (b-a)*x; }
```

11.46.1.8 __global__ void ZLUT_ComputeProfileMatchScores (int *njobs*, ZLUTParams *params*, float * *profiles*, float * *compareScoreBuf*, LocalizationParams * *locParams*)

```

114 {
115     int jobIdx = threadIdx.x + blockIdx.x * blockDim.x;
116     int zPlaneIdx = threadIdx.y + blockIdx.y * blockDim.y;
117
118     if (jobIdx >= njobs || zPlaneIdx >= params.planes)
119         return;
120
121     float* prof = &profiles [jobIdx * params.radialSteps()];
122     auto mapping = locParams[jobIdx];
123     float diffsum = 0.0f;
124     for (int r=0;r<params.radialSteps();r++) {
125         float d = prof[r] - params.img.pixel(r, zPlaneIdx, mapping.zlutIndex);
126         if (params.zcmpwindow)
127             d *= params.zcmpwindow[r];
128         diffsum += d*d;
129     }
130
131     compareScoreBuf[ params.planes * jobIdx + zPlaneIdx ] = -diffsum;
132 }
```

11.46.1.9 __global__ void ZLUT_ComputeZ (int *njobs*, ZLUTParams *params*, float3 * *positions*, float * *compareScoreBuf*)

```

101 {
102     int jobIdx = threadIdx.x + blockIdx.x * blockDim.x;
103
104     if (jobIdx < njobs) {
105         float* cmp = &compareScoreBuf [params.planes * jobIdx];
106
107         const float ZLUTFittingWeights[ZLUT_LSQFIT_NWEIGHTS] =
108             ZLUT_LSQFIT_WEIGHTS;
109         float maxPos = ComputeMaxInterp<float, ZLUT_LSQFIT_NWEIGHTS>::Compute
110             (cmp, params.planes, ZLUTFittingWeights);
111         positions[jobIdx].z = maxPos;
112     }
113 }
```

11.46.1.10 __global__ void ZLUT_NormalizeProfiles (int *njobs*, ZLUTParams *params*, float * *profiles*)

```

135 {
136     int jobIdx = threadIdx.x + blockIdx.x * blockDim.x;
137
138     if (jobIdx < njobs) {
139         float* prof = &profiles[params.radialSteps()*jobIdx];
140
141         // First, subtract mean
142         float mean = 0.0f;
143         for (int i=0;i<params.radialSteps();i++) {
144             mean += prof[i];
145         }
146         mean /= params.radialSteps();
147
148         float rmsSum2 = 0.0f;
149         for (int i=0;i<params.radialSteps();i++) {
150             prof[i] -= mean;
151             rmsSum2 += prof[i]*prof[i];
152         }
153
154         // And make RMS power equal 1
155         float invTotalRms = 1.0f / sqrt(rmsSum2/params.radialSteps());
156         for (int i=0;i<params.radialSteps();i++) {
157             prof[i] *= invTotalRms;
158         }
159 }
```

11.46.1.11 `__global__ void ZLUT_ProfilesToZLUT (int njobs, cudalmageListf images, ZLUTParams params, float3 * positions, LocalizationParams * locParams, float * profiles)`

```

61 {
62     int idx = threadIdx.x + blockDim.x * blockIdx.x;
63
64     if (idx < njobs) {
65         auto m = locParams[idx];
66         float* dst = params.GetRadialZLUT(m.zlutIndex, m.zlutPlane );
67
68         for (int i=0;i<params.radialSteps();i++)
69             dst [i] += profiles [ params.radialSteps()*idx + i ];
70     }
71 }
```

11.46.1.12 `template<typename TImageSampler> __global__ void ZLUT_RadialProfileKernel (int njobs, cudalmageListf images, ZLUTParams params, float3 * positions, float * profiles, float * means)`

```

76 {
77     int jobIdx = threadIdx.x + blockIdx.x * blockDim.x;
78     int radialIdx = threadIdx.y + blockIdx.y * blockDim.y;
79
80     if (jobIdx >= njobs || radialIdx >= params.radialSteps())
81         return;
82
83     float* dstprof = &profiles[params.radialSteps() * jobIdx];
84     float r = params.minRadius + (params.maxRadius-params.
85     minRadius)*radialIdx/params.radialSteps();
86     float sum = 0.0f;
87     int count = 0;
88
89     for (int i=0;i<params.angularSteps;i++) {
90         float x = positions[jobIdx].x + params.trigtable[i].x * r;
91         float y = positions[jobIdx].y + params.trigtable[i].y * r;
92
93         bool outside=false;
94         sum += TImageSampler::Interpolated(images, x,y, jobIdx, outside);
95         if (!outside) count++;
96     }
97     dstprof [radialIdx] = count>MIN_RADPROFILE_SMP_COUNT ? sum/count : means[jobIdx];
98 }
```

11.46.2 Variable Documentation

11.46.2.1 `surface<void, cudaSurfaceType2DLayered> image_lut_surface`

11.47 cudatrack/QI.h File Reference

Classes

- struct [QIParams](#)
- class [QI](#)
- struct [QI::StreamInstance](#)
- struct [QI::DeviceInstance](#)

TypeDefs

- typedef float [qivalue_t](#)
- typedef float2 [qicomplex_t](#)

11.47.1 Typedef Documentation

11.47.1.1 `typedef float2 qicomplex_t`

11.47.1.2 `typedef float qivalue_t`

11.48 cudatrack/QI_impl.h File Reference

```
#include "std_incl.h"
#include "utils.h"
#include "QueuedCUDATracker.h"
#include "ImageSampler.h"
#include "DebugResultCompare.h"
```

Functions

- template<typename TImageSampler>
`__device__ void ComputeQuadrantProfile (cudalImageListf &images, int idx, float *dst, const QIParams ¶ms, int quadrant, float2 center, float mean, int angularSteps)`
- template<typename TImageSampler>
`__global__ void QI_ComputeProfile (BaseKernelParams kp, float3 *positions, float *quadrants, float2 *profiles, float2 *reverseProfiles, const QIParams qiParams, float *d_radialweights, int angularSteps)`
- `__global__ void QI_MultiplyWithConjugate (int n, cufftComplex *a, cufftComplex *b)`
- `__device__ float QI_ComputeAxisOffset (cufftComplex *autoconv, int fftlen, float *shiftbuf)`
- `__global__ void QI_OffsetPositions (int njobs, float3 *current, float3 *dst, cufftComplex *autoconv, int fftLength, float2 *offsets, float pixelsPerProfLen, float *shiftbuf)`
- template<typename TImageSampler>
`__global__ void QI_ComputeQuadrants (BaseKernelParams kp, float3 *positions, float *dst_quadrants, const QIParams *params, int angularSteps)`
- `__global__ void QI_QuadrantsToProfiles (BaseKernelParams kp, float *quadrants, float2 *profiles, float2 *reverseProfiles, float *d_radialweights, const QIParams *params)`

11.48.1 Function Documentation

11.48.1.1 template<typename TImageSampler> __device__ void ComputeQuadrantProfile (`cudalImageListf & images, int idx, float * dst, const QIParams & params, int quadrant, float2 center, float mean, int angularSteps`)

```
9 {
10     const int qmat[] = {
11         1, 1,
12         -1, 1,
13         -1, -1,
14         1, -1 };
15     int mx = qmat[2*quadrant+0];
16     int my = qmat[2*quadrant+1];
17
18     //for (int i=0;i<params.radialSteps;i++)
19     //  dst[i]=0.0f;
20
21     float asf = (float)params.trigtablesize / angularSteps;
22     float rstep = (params.maxRadius - params.minRadius) / params.
23     radialSteps;
24     for (int i=0;i<params.radialSteps; i++) {
25         float sum = 0.0f;
26         float r = params.minRadius + rstep * i;
27         int count=0;
```

```

28     for (int a=0;a<angularSteps;a++) {
29         int j = (int)(asf * a);
30         float x = center.x + mx*params.cos_sin_table[j].x * r;
31         float y = center.y + my*params.cos_sin_table[j].y * r;
32         bool outside=false;
33         float v = TImageSampler::Interpolated(images, x,y, idx, outside);
34         if (!outside) {
35             sum += v;
36             count++;
37         }
38     }
39     dst[i] = count >= MIN_RADPROFILE_SMP_COUNT ? sum/count : mean;
40 }
41 }
42 }
```

11.48.1.2 `__device__ float QI_ComputeAxisOffset(cufftComplex * autoconv, int fftlen, float * shiftbuf)`

```

112 {
113     typedef float compute_t;
114     int nr = fftlen/2;
115     for(int x=0;x<fftlen;x++) {
116         shiftbuf[x] = autoconv[ (x+nr)%(nr*2) ].x;
117     }
118
119     const float QIWeights[QI_LSQFIT_NWEIGHTS] =
120     QI_LSQFIT_WEIGHTS;
121
122     compute_t maxPos = ComputeMaxInterp<compute_t>::Compute(shiftbuf,
123     fftlen, QIWeights);
124     compute_t offset = (maxPos - nr) * (3.14159265359f / 4);
125     return offset;
126 }
```

11.48.1.3 template<typename TImageSampler> `__global__ void QI_ComputeProfile(BaseKernelParams kp, float3 * positions, float * quadrants, float2 * profiles, float2 * reverseProfiles, const QIParams qiParams, float * d_radialweights, int angularSteps)`

```

46 {
47     int idx = threadIdx.x + blockDim.x * blockIdx.x;
48     if (idx < kp.njobs) {
49         const QIParams& qp = qiParams;
50         int fftlen = qp.radialSteps*2;
51         float* img_qdr = &quadrants[ idx * qp.radialSteps * 4 ];
52         for (int q=0;q<4;q++) {
53             ComputeQuadrantProfile<TImageSampler>( kp.images, idx, &img_qdr[q*qp.
54             radialSteps], qp, q,
55             make_float2(positions[idx].x, positions[idx].y), kp.imgmeans[idx], angularSteps );
56         }
57         int nr = qp.radialSteps;
58         qicomplex_t* imgprof = (qicomplex_t*)&profiles[idx * fftlen*2];
59         qicomplex_t* x0 = imgprof;
60         qicomplex_t* x1 = imgprof + nr*1;
61         qicomplex_t* y0 = imgprof + nr*2;
62         qicomplex_t* y1 = imgprof + nr*3;
63
64         qicomplex_t* revprof = (qicomplex_t*)&reverseProfiles[idx*fftlen*2];
65         qicomplex_t* xrev = revprof;
66         qicomplex_t* yrev = revprof + nr*2;
67
68         float* q0 = &img_qdr[0];
69         float* q1 = &img_qdr[nr];
70         float* q2 = &img_qdr[nr*2];
71         float* q3 = &img_qdr[nr*3];
72
73         // Build Ix = qL(-r) || qR(r)
74         // qL = q1 + q2 (concat0)
75         // qR = q0 + q3 (concat1)
76         for(int r=0;r<nr;r++) {
77             float rw = d_radialweights[r];
78             x0[nr-r-1] = make_float2(rw*(q1[r]+q2[r]), 0);
79             x1[r] = make_float2( rw*(q0[r]+q3[r]),0 );
80         }
81
82         // Build Iy = [ qB(-r) qT(r) ]
```

```

83     // qT = q0 + q1
84     // qB = q2 + q3
85     for(int r=0;r<nr;r++) {
86         float rw = d_radialweights[r];
87         y1[r] = make_float2( rw * ( q0[r]+q1[r] ),0 );
88         y0[nr-r-1] = make_float2( rw * ( q2[r]+q3[r] ),0 );
89     }
90
91     for(int r=0;r<nr*2;r++) {
92         xrev[r] = x0[nr*2-r-1];
93         for(int r=0;r<nr*2;r++) {
94             yrev[r] = y0[nr*2-r-1];
95         }
96     }

```

11.48.1.4 template<typename TImageSampler> __global__ void QI_ComputeQuadrants (BaseKernelParams kp, float3 * positions, float * dst_quadrants, const QIParams * params, int angularSteps)

```

159 {
160     int jobIdx = threadIdx.x + blockIdx.x * blockDim.x;
161     int rIdx = threadIdx.y + blockIdx.y * blockDim.y;
162     int quadrant = threadIdx.z;
163
164     // Ori: dst[i], i = radial index
165     // float* img_qdr = &dst_quadrants[ jobIdx * params->radialSteps * 4 ];
166     // float* dst = &img_qdr[quadrant*params->radialSteps];
167     // dst[rIdx] = rIdx;//count >= MIN_RADPROFILE_SMP_COUNT ? sum/count : kp.imgmeans[jobIdx];
168
169     if (jobIdx < kp.njobs && rIdx < params->radialSteps && quadrant < 4) {
170         // The variables below could go in shared memory
171         float asf = (float)params->trigtablesize / angularSteps;
172         float rstep = (params->maxRadius - params->minRadius) / params->
173         radialSteps;
174         const int qmat[] = {
175             1, 1,
176             -1, 1,
177             -1, -1,
178             1, -1 };
179
180         // --Stop--
181         int mx = qmat[2*quadrant+0];
182         int my = qmat[2*quadrant+1];
183
184         // Ori: dst[i], i = radial index
185         // float* img_qdr = &quadrants[ idx * qp.radialSteps * 4 ];
186         // dst = &img_qdr[q*qp.radialSteps]
187         float* qdr = &dst_quadrants[ (jobIdx * 4 + quadrant) * params->
188         radialSteps ];
189
190         float sum = 0.0f;
191         float r = params->minRadius + rstep * rIdx;
192         float3 pos = positions[jobIdx];
193         float mean = imgmeans[jobIdx];
194
195         int count=0;
196         for (int a=0;a<angularSteps;a++) {
197             int j = (int)(asf * a);
198             float x = pos.x + mx*params->cos_sin_table[j].x * r;
199             float y = pos.y + my*params->cos_sin_table[j].y * r;
200             bool outside=false;
201             float v = TImageSampler::Interpolated(kp.images, x,y,jobIdx, outside);
202             if (!outside) {
203                 sum += v;
204                 count++;
205             }
206             qdr[rIdx] = count >= MIN_RADPROFILE_SMP_COUNT ? sum/count : kp.
207             imgmeans[jobIdx];
208         }

```

11.48.1.5 __global__ void QI_MultiplyWithConjugate (int n, cufftComplex * a, cufftComplex * b)

```

99 {
100     //int idx = (threadIdx.y + threadIdx.x << 4) + (blockIdx.x + blockIdx.y *( (int)sqrt( (double)(n /

```

```

101    (blockDim.x * blockDim.y)) ) + 1)) << 8;
102    //int idx = (threadIdx.x + blockIdx.y * blockDim.x) + (blockIdx.x + blockIdx.y *( (int)sqrt(
103    (double)(n / (blockDim.x * blockDim.y)) ) + 1)) * blockDim.x * blockDim.y;
104    int idx = threadIdx.x + blockIdx.x * blockDim.x;
105    if (idx < n) {
106        cufftComplex A = a[idx];
107        cufftComplex B = b[idx];
108    }
109 }

```

11.48.1.6 __global__ void QI_OffsetPositions (int njobs, float3 * current, float3 * dst, cufftComplex * autoconv, int fftLength, float2 * offsets, float pixelsPerProfLen, float * shiftbuf)

```

127 {
128     int idx = threadIdx.x + blockIdx.x * blockDim.x;
129
130     if (idx < njobs) {
131         float* shifted = &shiftbuf[ idx * fftLength ];
132
133         // X
134         cufftComplex* autoconvX = &autoconv[idx * fftLength * 2];
135         float xoffset = QI_ComputeAxisOffset(autoconvX, fftLength, shifted);
136
137         cufftComplex* autoconvY = autoconvX + fftLength;
138         float yoffset = QI_ComputeAxisOffset(autoconvY, fftLength, shifted);
139
140         dst[idx].x = current[idx].x + xoffset * pixelsPerProfLen;
141         dst[idx].y = current[idx].y + yoffset * pixelsPerProfLen;
142         dst[idx].z = current[idx].z;
143
144         if (offsets)
145             offsets[idx] = make_float2( xoffset, yoffset );
146     }
147 }

```

11.48.1.7 __global__ void QI_QuadrantsToProfiles (BaseKernelParams kp, float * quadrants, float2 * profiles, float2 * reverseProfiles, float * d_radialweights, const QIParams * params)

```

211 {
212     //ComputeQuadrantProfile(cudaImageListf& images, int idx, float* dst, const QIParams& params, int quadrant,
213     //float2 center)
214     int idx = threadIdx.x + blockDim.x * blockIdx.x;
215     if (idx < kp.njobs) {
216         int fftlen = params->radialSteps*2;
217         float* img_qdr = &quadrants[ idx * params->radialSteps * 4 ];
218         // for (int q=0;q<4;q++)
219             //ComputeQuadrantProfile<TImageSampler> (images, idx, &img_qdr[q*params->radialSteps], params,
220             q, img_means[idx], make_float2(positions[idx].x, positions[idx].y));
221
222         int nr = params->radialSteps;
223         qicomplex_t* imgprof = (qicomplex_t*) &profiles[idx * fftlen*2];
224         qicomplex_t* x0 = imgprof;
225         qicomplex_t* x1 = imgprof + nr*1;
226         qicomplex_t* y0 = imgprof + nr*2;
227         qicomplex_t* y1 = imgprof + nr*3;
228
229         qicomplex_t* revprof = (qicomplex_t*)&reverseProfiles[idx*fftlen*2];
230         qicomplex_t* xrev = revprof;
231         qicomplex_t* yrev = revprof + nr*2;
232
233         float* q0 = &img_qdr[0];
234         float* q1 = &img_qdr[nr];
235         float* q2 = &img_qdr[nr*2];
236         float* q3 = &img_qdr[nr*3];
237
238         // Build Ix = qL(-r) || qR(r)
239         // qL = q1 + q2 (concat0)
240         // qR = q0 + q3 (concat1)
241         for(int r=0;r<nr;r++) {
242             float rw = d_radialweights[r];
243             x0[nr-r-1] = make_float2( rw * (q1[r]+q2[r]), 0 );
244             x1[r] = make_float2( rw * (q0[r]+q3[r]), 0 );
245         }
246         // Build Iy = [ qB(-r) qT(r) ]

```

```

245     // qT = q0 + q1
246     // qB = q2 + q3
247     for(int r=0;r<nr;r++) {
248         float rw = d_radialweights[r];
249         y1[r] = make_float2( rw * (q0[r]+q1[r]),0);
250         y0[nr-r-1] = make_float2( rw * (q2[r]+q3[r]),0);
251     }
252
253     for(int r=0;r<nr*2;r++) {
254         xrev[r] = x0[nr*2-r-1];
255         for(int r=0;r<nr*2;r++)
256             yrev[r] = y0[nr*2-r-1];
257     }
258 }
```

11.49 cudatrack/QueuedCUDATracker.cu File Reference

```

#include "std_incl.h"
#include <algorithm>
#include "cuda_runtime.h"
#include "device_launch_parameters.h"
#include "vector_types.h"
#include <cstdint>
#include "utils.h"
#include "cpu_tracker.h"
#include "QueuedCUDATracker.h"
#include "gpu_utils.h"
#include "ImageSampler.h"
#include "Kernels.h"
#include "DebugResultCompare.h"
#include "QI_impl.h"
```

Classes

- class [ScopedCPUProfiler](#)

Macros

- [#define TRK_PROFILE](#)

Functions

- void [SetCUDADevices](#) (**int** *dev, **int** numdev)
- [QueuedTracker](#) * [CreateQueuedTracker](#) (**const QTrkComputedConfig** &cc)
- static **int** [GetBestCUDADevice](#) ()
- **__global__** void [AddProfilesToZLUT](#) (**float** *d_src, **int** nbeads, **int** radialsteps, **int** plane, [cudaImageListf](#) zlut)
- template<**typename T**>
 void [checksum](#) (**T** *data, **int** elemsize, **int** numelem, **const char** *name)

Variables

- static std::vector<**int**> [cudaDeviceList](#)

11.49.1 Macro Definition Documentation

11.49.1.1 #define TRK_PROFILE

11.49.2 Function Documentation

11.49.2.1 __global__ void AddProfilesToZLUT (float * d_src, int nbeads, int radialsteps, int plane, cudalmageListf zlut)

```

459 {
460     int b = threadIdx.x + blockIdx.x * blockDim.x;
461     int r = threadIdx.y + blockIdx.y * blockDim.y;
462
463     if (b < nbeads && r < radialsteps) {
464         zlut.pixel(r, plane, b) += d_src[ b * radialsteps + r ];
465     }
466 }
```

11.49.2.2 template<typename T> void checksum (T * data, int elemsize, int numelem, const char * name)

```

571 {
572 #ifdef QI_DBG_EXPORT
573     uchar* cp = (uchar*)ALLOCA(elemsize*numelem*sizeof(T));
574     cudaDeviceSynchronize();
575     cudaMemcpy(cp, data, sizeof(T)*elemsize*numelem, cudaMemcpyDeviceToHost);
576
577     dbgprintf("%s:\n", name);
578     for (int i=0;i<numelem;i++) {
579         uchar *elem = cp+elemsize*sizeof(T)*i;
580         dbgprintf("[%d]: %d\n", i, hash(elem, elemsize));
581     }
582 #endif
583 }
```

11.49.2.3 QueuedTracker* CreateQueuedTracker (const QTrkComputedConfig & cc)

```

89 {
90     return new QueuedCUDATracker(cc);
91 }
```

11.49.2.4 static int GetBestCUDADevice () [static]

```

94 {
95     int bestScore;
96     int bestDev;
97     int numDev;
98     cudaGetDeviceCount(&numDev);
99     for (int a=0;a<numDev;a++) {
100         int score;
101         cudaDeviceProp prop;
102         cudaGetDeviceProperties(&prop, a);
103         score = prop.multiProcessorCount * prop.clockRate;
104         if (a==0 || bestScore < score) {
105             bestScore = score;
106             bestDev = a;
107         }
108     }
109     return bestDev;
110 }
```

11.49.2.5 void SetCUDADevices (int * dev, int numdev)

```
84     cudaDeviceList.assign(dev, dev+numdev);  
85 }  
86 }
```

11.49.3 Variable Documentation

11.49.3.1 std::vector<int> cudaDeviceList [static]

11.50 cudatrack/QueuedCUDATracker.h File Reference

```
#include "QueuedTracker.h"  
#include "threads.h"  
#include <cuda_runtime_api.h>  
#include <cufft.h>  
#include <list>  
#include <vector>  
#include <map>  
#include "gpu_utils.h"  
#include "cudaImageList.h"  
#include "QI.h"
```

Classes

- struct [cudaImageList< T >](#)
- struct [LocalizationParams](#)
- struct [BaseKernelParams](#)
- struct [ZLUTParams](#)
- struct [ImageLUTConfig](#)
- class [QueuedCUDATracker](#)
CUDA implementation of the [QueuedTracker](#) interface.
- struct [QueuedCUDATracker::Device](#)
- struct [QueuedCUDATracker::Stream](#)
- struct [QueuedCUDATracker::KernelProfileTime](#)

TypeDefs

- typedef [cudaImageList< float > cudaImageListf](#)

11.50.1 Typedef Documentation

11.50.1.1 typedef [cudaImageList<float> cudaImageListf](#)

11.51 cudatrack/simplefft.h File Reference

```
#include <complex>
```

Classes

- struct `sfft::complex< T >`

Namespaces

- `sfft`

Macros

- `#define SFFT_BOTH __device__ __host__`

Functions

- template<typename T >
`SFFT_BOTH void sfft::swap (T &a, T &b)`
- template<typename T , int sign>
`SFFT_BOTH void sfft::fft (size_t N, std::complex< T > *zs, complex< T > *twiddles)`
- template<typename T >
`std::vector< complex< T > > sfft::fill_twiddles (int N)`
- template<typename T >
`SFFT_BOTH void sfft::fft_forward (size_t N, complex< T > *zs, complex< T > *twiddles)`
- template<typename T >
`SFFT_BOTH void sfft::fft_inverse (size_t N, complex< T > *zs, complex< T > *twiddles)`

11.51.1 Macro Definition Documentation

11.51.1.1 `#define SFFT_BOTH __device__ __host__`

Index

_CRT_SECURE_NO_WARNINGS
 std_incl.h, 321

_STDINT_H
 labview.h, 294

_inverse
 kissfft, 110

_nfft
 kissfft, 110

_stageRadix
 kissfft, 110

_stageRemainder
 kissfft, 110

_twiddles
 kissfft, 110

~ClImageData
 ClImageData, 43

~CPUTracker
 CPUTracker, 52

~Device
 QueuedCUDATracker::Device, 75

~DeviceInstance
 QI::DeviceInstance, 81

~FFT2D
 CPUTracker::FFT2D, 82

~FFT2DTracker
 FFT2DTracker, 84

~Image4DCudaArray
 Image4DCudaArray, 89

~Image4DMemory
 Image4DMemory, 94

~Job
 QueuedCPUTracker::Job, 100

~MeasureTime
 MeasureTime, 130

~Mutex
 Threads::Mutex, 131

~QueuedCPUTracker
 QueuedCPUTracker, 156

~QueuedCUDATracker
 QueuedCUDATracker, 175

~QueuedTracker
 QueuedTracker, 190

~ResultFile
 ResultFile, 197

~ResultManager
 ResultManager, 198

~ScopedCPUProfiler
 ScopedCPUProfiler, 210

~Stream
 QueuedCUDATracker::Stream, 213

~StreamInstance
 QI::StreamInstance, 216

~ThreadPool
 ThreadPool, 220

~device_vec
 device_vec, 78

~outputter
 outputter, 134

~pinned_array
 pinned_array, 137

a
 LsqSqQuadFit, 116

ALLOCA_ARRAY
 std_incl.h, 321

ALLOCA
 std_incl.h, 321

ANGSTEPF
 main.cpp, 265

API - C, 29
 QTrkBuildLUT, 29
 QTrkClearResults, 29
 QTrkCreateInstance, 30
 QTrkFetchResults, 30
 QTrkFinalizeLUT, 30
 QTrkFlush, 30
 QTrkFreeInstance, 30
 QTrkGetComputedConfig, 30
 QTrkGetProfileReport, 31
 QTrkGetQueueLength, 31
 QTrkGetRadialZLUTSize, 31
 QTrkGetRadialZLUT, 31
 QTrkGetResultCount, 31
 QTrkGetWarnings, 31
 QTrkIsIdle, 31
 QTrkScheduleFrame, 32
 QTrkScheduleLocalization, 32
 QTrkSetLocalizationMode, 32
 QTrkSetRadialZLUT, 32

API - LabVIEW, 23
 QFF_Force32Bit, 24
 qtrk_clear_results, 24
 qtrk_create, 24
 qtrk_destroy, 24
 qtrk_dump_memleaks, 24
 qtrk_flush, 25
 qtrk_generate_image_from_lut, 25
 qtrk_get_ZLUT, 25
 qtrk_get_profile_report, 25

qtrk_get_results, 25
 qtrk_idle, 26
 qtrk_queue_array, 26
 qtrk_queue_float, 26
 qtrk_queue_frame, 26
 qtrk_queue_pitchedmem, 26
 qtrk_queue_u16, 27
 qtrk_queue_u8, 27
 qtrk_read_timestamp, 27
 qtrk_resultcount, 27
 qtrk_set_ZLUT, 27
 qtrkcuda_device_count, 28
 qtrkcuda_set_device_list, 28
 QueueFrameFlags, 24

abc
 LsqSqQuadFit::Coeff, 45

abs
 BeadFinder.cpp, 282

acc
 SpeedAccResult, 211

AccBiasTest
 main.cpp, 242

AddImages
 testutils.cpp, 274
 testutils.h, 278

AddJob
 QueuedCPUTracker, 156

AddProfilesToZLUT
 QueuedCUDATracker.cu, 382

AddWork
 ThreadPool, 220

alloc
 cudalImageList, 70
 TImageData, 225

allocateHostImageBuffer
 cudalImageList, 70

AllocateJob
 QueuedCPUTracker, 156

AllocateQIFFTs
 CPUTracker, 52

angularSteps
 ZLUTParams, 235

Apply
 CPUTracker::FFT2D, 83

ApplyGaussianNoise
 utils.cpp, 324
 utils.h, 337

ApplyOffsetGain
 CPUTracker, 52
 Kernels.h, 372
 QueuedCPUTracker, 156

ApplyPoissonNoise
 utils.cpp, 324
 utils.h, 337

ArgumentErrorMsg
 LabVIEW datatypes and helper functions, 20

array
 Image4DCudaArray, 93

at
 Matrix3X3, 127
 TImageData, 225

Atomic
 Atomic, 37
 data, 38
 get, 37
 m, 38
 operator T, 37
 operator=, 38
 set, 38

Atomic< T >, 37

AutoBeadFindTest
 main.cpp, 243

b
 LsqSqQuadFit, 116

BINFILE_VERSION
 ResultManager.cpp, 319

BUILDLUT_BIASCORRECT
 QueuedTracker.h, 317

BUILDLUT_FOURIER
 QueuedTracker.h, 317

BUILDLUT_IMAGELUT
 QueuedTracker.h, 317

BUILDLUT_NORMALIZE
 QueuedTracker.h, 317

Backg
 main.cpp, 242

BackgroundMedian
 testutils.cpp, 274
 testutils.h, 278

BackgroundRMS
 testutils.cpp, 274
 testutils.h, 278

BackgroundStdDev
 testutils.cpp, 274
 testutils.h, 278

BaseKernelParams, 38
 images, 38
 imgmeans, 38
 locParams, 38
 njobs, 38

BasicQTrkTest
 test.cu, 353

BasicQTrkTest_RM
 test.cu, 354

batchSize
 QI, 144
 QueuedCUDATracker, 187

batchStart
 QueuedCUDATracker::Stream, 214

batchesDone
 QueuedCUDATracker, 187

BeadFinder, 33
 Find, 33, 34

BeadFinder.cpp
 abs, 282
 ComplexToJPEGFile, 282

FFT2D, 282
NextPowerOf2, 282
RecenterAndFilter, 282
SearchArea, 283
BeadFinder::Config, 49
 img_distance, 49
 MinPixelDistance, 49
 roi, 49
 similarity, 49
BeadFinder::Position, 140
 Position, 140
 x, 140
 y, 140
begin
 pinned_array, 138
BeginLUT
 QueuedCPUTracker, 157
 QueuedCUDATracker, 175
 QueuedTracker, 190
Benchmark.cpp
 BenchmarkConfigParamRange, 238
 BenchmarkParams, 238
 BenchmarkROISizes, 239
 BenchmarkZAccuracy, 239
 ElectronsPerBit, 240
 img_mean, 240
 img_sigma, 240
BenchmarkConfigParamRange
 Benchmark.cpp, 238
BenchmarkLUT, 39
 BenchmarkLUT, 39
 CleanupLUT, 40
 GenerateLUT, 40
 GenerateSample, 40
 Load, 40, 41
 lut_h, 41
 lut_w, 41
 max_a, 41
 max_b, 41
 max_c, 41
 normprof, 41
BenchmarkParams
 Benchmark.cpp, 238
 main.cpp, 243
 test.cu, 355
BenchmarkROISizes
 Benchmark.cpp, 239
BenchmarkZAccuracy
 Benchmark.cpp, 239
bg
 CPUTracker::Gauss2DResult, 87
BgCorrectedCOM
 Kernels.h, 372, 373
bias
 SpeedAccResult, 211
binaryOutput
 ResultManagerConfig, 205
BinaryResultFile, 41
BinaryResultFile, 42
f, 42
LoadRow, 42
SaveRow, 42
bind
 cudaImageList, 70
 Image4DCudaArray, 90
 Image4DMemory, 94
BindTexture
 ImageSampler_InterpolatedTexture, 97
 ImageSampler_MemCopy, 98
 ImageSampler_SimpleTextureRead, 99
blocks
 QI, 141
 QueuedCUDATracker, 175
boundaryHit
 cudaImageList, 70
Break
 QueuedCPUTracker, 157
BuildConvergenceMap
 main.cpp, 244
BuildLUT
 QueuedCPUTracker, 157
 QueuedCUDATracker, 175
 QueuedTracker, 190
BuildZLUT
 main.cpp, 245
 test.cu, 356
c
 LsqSqQuadFit, 116
C_ADDTO
 kissfft, 105
C_ADD
 kissfft, 105
C_FIXDIV
 kissfft, 106
C_MULBYSCALAR
 kissfft, 106
C_MUL
 kissfft, 106
C_SUB
 kissfft, 106
CDLL_EXPORT
 dllmacros.h, 289
CImageData, 43
 ~CImageData, 43
 CImageData, 43
 operator=, 44
CPU_ApplyOffsetGain
 QueuedCUDATracker, 177
CPUTracker, 50
 ~CPUTracker, 52
 AllocateQIFFTs, 52
 ApplyOffsetGain, 52
 CPUTracker, 52
 CalculateErrorCurve, 53
 CalculateErrorFlag, 53
 CalculateInterpolatedZLUTProfile, 53

CheckBoundaries, 54
 Compute2DGaussianMLE, 54
 ComputeAsymmetry, 55
 ComputeMeanAndCOM, 55
 ComputeQI, 56
 ComputeQuadrantProfile, 57
 ComputeRadialProfile, 58
 ComputeXCorInterpolated, 58
 ComputeZ, 59
 debugImage, 66
 fft2d, 66
 FourierRadialProfile, 59
 FourierTransform2D, 60
 GetDebugImage, 60
 GetHeight, 60
 GetPixel, 60
 GetRadialZLUT, 60
 GetWidth, 60
 height, 66
 KeepInsideBoundaries, 60
 LUTProfMaxQuadraticFit, 51
 LUTProfMaxSimpleInterp, 51
 LUTProfMaxSplineFit, 51
 LUTProfileCompare, 61
 LUTProfileCompareAdjustedWeights, 63
 LUTProfileMaxComputeMode, 51
 mean, 66
 Normalize, 63
 QI_ComputeOffset, 63
 qa_fft_backward, 67
 qa_fft_forward, 67
 qi_fft_backward, 67
 qi_fft_forward, 67
 qi_radialsteps, 67
 QuadrantAlign, 64
 QuadrantAlign_ComputeOffset, 65
 quadrantDirs, 67
 radialDirs, 67
 SaveImage, 65
 SetImage, 65
 SetImage16Bit, 65
 SetImage8Bit, 65
 SetImageFloat, 66
 SetRadialWeights, 66
 SetRadialZLUT, 66
 srclImage, 67
 stdev, 67
 testRun, 67
 trackerID, 67
 width, 67
 xcorBuffer, 67
 xcorw, 67
 zlut_count, 67
 zlut_maxradius, 67
 zlut_memoryOwner, 67
 zlut_minradius, 67
 zlut_planes, 67
 zlut_radialweights, 67
 zlut_res, 67
 zlut_useCorrelation, 67
 zluts, 67
 CPUTracker::FFT2D, 82
 ~FFT2D, 82
 Apply, 83
 cbuf, 83
 FFT2D, 82
 xfft, 83
 yfft, 83
 CPUTracker::Gauss2DResult, 86
 bg, 87
 I0, 87
 pos, 87
 CRP_TestGeneratedData
 main.cpp, 247
 CUBOTH
 gpu_utils.h, 369
 CUDA_SUPPORTED_FUNC
 CubicBSpline.h, 287
 gpu_utils.h, 369
 LsqQuadraticFit.h, 294
 CUDADeviceInfo, 68
 clockRate, 68
 major, 68
 minor, 68
 multiProcCount, 68
 name, 68
 calculate
 LsqSqQuadFit, 115
 CalculateErrorCurve
 CPUTracker, 53
 CalculateErrorFlag
 CPUTracker, 53
 CalculateInterpolatedZLUTProfile
 CPUTracker, 53
 calib_gain
 QueuedCPUTracker, 170
 QueuedCUDATracker::Device, 76
 calib_offset
 QueuedCPUTracker, 170
 QueuedCUDATracker::Device, 76
 callback
 Threads::Handle, 87
 capacity
 cudaImageList, 70
 capturedFrames
 ResultManager::FrameCounters, 85
 cbuf
 CPUTracker::FFT2D, 83
 cfg
 QueuedTracker, 196
 check_arg
 test.cu, 357
 check_strarg
 test.cu, 357
 CheckBoundaries
 CPUTracker, 54

CheckCUDAError
 gpu_utils.h, 369
CheckImageInput
 lv_queuetrk_api.cpp, 302
CheckResultSpace
 ResultManager, 199
checksum
 QueuedCUDATracker.cu, 382
clamp
 cpu_tracker.cpp, 285
CleanupLUT
 BenchmarkLUT, 40
clear
 cudaImageList, 70
 Image4DCudaArray, 90
 Image4DMemory, 94
ClearResults
 QueuedCPUTracker, 158
 QueuedCUDATracker, 176
 QueuedTracker, 190
clockRate
 CUDADeviceInfo, 68
CmdLineRun
 test.cu, 357
cmp_cpu_qi_fft_out
 test.cu, 368
cmp_cpu_qi_prof
 test.cu, 368
cmp_gpu_qi_fft_out
 test.cu, 368
cmp_gpu_qi_prof
 test.cu, 368
cnt
 ResultManager, 204
code
 ErrorCluster, 82
 LVDataType< double >, 120
 LVDataType< float >, 121
 LVDataType< int16_t >, 121
 LVDataType< int32_t >, 122
 LVDataType< int64_t >, 122
 LVDataType< int8_t >, 123
 LVDataType< std::complex< double > >, 123
 LVDataType< std::complex< float > >, 124
 LVDataType< uint16_t >, 124
 LVDataType< uint32_t >, 125
 LVDataType< uint64_t >, 125
 LVDataType< uint8_t >, 126
cols
 Image4DMemory::KernelParams, 103
com
 QueuedCUDATracker::KernelProfileTime, 104
 QueuedCUDATracker::Stream, 214
com_bgcorrection
 QTrkSettings, 150
comDone
 QueuedCUDATracker::Stream, 214
CompareAccuracy
 test.cu, 359
complex
 sfft::complex, 46
complex_t
 scalar_types.h, 320
ComplexToJPEGFile
 BeadFinder.cpp, 282
Compute
 ComputeMaxInterp, 48
 SampleFisherMatrix, 208
 SpeedAccResult, 211
compute
 LsqSqQuadFit, 115
 test.cu, 360
Compute2DGaussianMLE
 CPUTracker, 54
compute_asymmetry
 lv_cputrack_api.cpp, 295
compute_com
 lv_cputrack_api.cpp, 295
compute_crp
 lv_cputrack_api.cpp, 295
compute_qi
 lv_cputrack_api.cpp, 296
compute_radial_profile
 lv_cputrack_api.cpp, 296
compute_xcor
 lv_cputrack_api.cpp, 296
compute_z
 lv_cputrack_api.cpp, 296
ComputeAsymmetry
 CPUTracker, 55
ComputeAverageFisher
 SampleFisherMatrix, 208
ComputeBSplineDerivatives
 CubicBSpline.h, 287
ComputeBSplineWeights
 CubicBSpline.h, 287
ComputeBgCorrectedCOM1D
 utils.cpp, 325
 utils.h, 337
ComputeCRP
 utils.cpp, 325
 utils.h, 338
computeDeriv
 LsqSqQuadFit, 115
computeImagePos
 cudaImageList, 70
ComputeMax2DInterpolated
 FFT2DTracker, 84
ComputeMaxInterp
 Compute, 48
 max_, 48
 min_, 49
ComputeMaxInterp< T, numPts >, 48
ComputeMeanAndCOM
 CPUTracker, 55
ComputeQI

CPUTracker, 56
 ComputeQuadrantProfile
 CPUTracker, 57
 QI_impl.h, 377
 ComputeRadialBinWindow
 utils.cpp, 326
 utils.h, 338
 ComputeRadialProfile
 CPUTracker, 58
 utils.cpp, 326
 utils.h, 339
 ComputeRadialWeights
 main.cpp, 246
 ComputeSplineFitMaxPos
 CubicBSpline.h, 288
 computeStats
 RunTrackerResults, 206
 ComputeStdDev
 utils.h, 339
 computeSums
 LsqSqQuadFit, 115
 ComputeXCor
 FFT2DTracker, 84
 ComputeXCorInterpolated
 CPUTracker, 58
 ComputeZBiasCorrection
 QueuedTracker, 190
 ComputeZ
 CPUTracker, 59
 Config
 ResultManager, 199
 config
 ResultManager, 204
 ConfigValueMap
 QueuedTracker, 190
 conjugate
 cpu_tracker.cpp, 285
 sfft::complex, 46
 Console
 outputter::outputModes, 133
 testutils.h, 277
 copyImageToDevice
 cudaImageList, 71
 Image4DCudaArray, 90
 Image4DMemory, 94
 CopyImageToFloat
 QueuedTracker.h, 317
 utils.cpp, 327
 copyImageToHost
 cudaImageList, 71
 Image4DCudaArray, 91
 Image4DMemory, 94
 CopyStreamResults
 QueuedCUDATracker, 176
 copyTo
 TImageData, 225
 copyToDevice
 cudaImageList, 71
 device_vec, 78
 Image4DCudaArray, 91
 Image4DMemory, 95
 copyToHost
 cudaImageList, 71
 device_vec, 79
 Image4DCudaArray, 91
 Image4DMemory, 95
 CorrectedRadialProfileTest
 main.cpp, 246
 cos_sin_table
 QIParams, 144
 count
 cudaImageList, 74
 ResultManager::FrameResult, 86
 cpu_time
 QueuedCUDATracker, 187
 cpu_tracker.cpp
 clamp, 285
 conjugate, 285
 MARKPIXELI, 285
 MARKPIXEL, 285
 QIWeights, 286
 round, 286
 SFFT_BOTH, 285
 sum_diff, 286
 XCorScale, 286
 ZLUT_CMPDATA, 285
 ZLUTWeights, 286
 ZLUTWeights_d, 286
 cpu_tracker.h
 CreateCPUTrackerInstance, 287
 pixel_t, 287
 cputrack-test/Benchmark.cpp, 237
 cputrack-test/SharedTests.h, 266
 cputrack-test/main.cpp, 240
 cputrack-test/testutils.cpp, 273
 cputrack-test/testutils.h, 277
 cputrack/BeadFinder.cpp, 281
 cputrack/BeadFinder.h, 284
 cputrack/BenchmarkLUT.cpp, 284
 cputrack/BenchmarkLUT.h, 284
 cputrack/CubicBSpline.h, 287
 cputrack/DebugResultCompare.h, 288
 cputrack/FFT2DTracker.h, 292
 cputrack/FisherMatrix.h, 292
 cputrack/LsqQuadraticFit.h, 294
 cputrack/QueuedCPUTracker.cpp, 315
 cputrack/QueuedCPUTracker.h, 315
 cputrack/QueuedTracker.cpp, 316
 cputrack/QueuedTracker.h, 316
 cputrack/ResultManager.cpp, 319
 cputrack/ResultManager.h, 319
 cputrack/cpu_tracker.cpp, 285
 cputrack/cpu_tracker.h, 286
 cputrack/dllmacros.h, 288
 cputrack/fastjpg.cpp, 289
 cputrack/hash_templates.h, 292

cuptrack/kissfft.h, 293
cuptrack/labview.h, 293
cuptrack/lv_cputrack_api.cpp, 294
cuptrack/lv_qtrk_api.h, 299
cuptrack/lv_queuetrk_api.cpp, 300
cuptrack/lv_resultmanager_api.cpp, 308
cuptrack/memdbg.cpp, 311
cuptrack/memdbg.h, 311
cuptrack/qtrk_c_api.cpp, 311
cuptrack/qtrk_c_api.h, 312
cuptrack/random_distr.h, 318
cuptrack/scalar_types.h, 320
cuptrack/std_incl.h, 320
cuptrack/threads.h, 322
cuptrack/utils.cpp, 323
cuptrack/utils.h, 335
cpx_type
 kissfft, 105
 kissfft_utils::traits, 227
Create
 Threads, 223
create_tracker
 lv_cputrack_api.cpp, 297
CreateCPUTrackerInstance
 cpu_tracker.h, 287
CreateQueuedTracker
 QueuedCPUTracker.cpp, 315
 QueuedCUDATracker.cu, 382
 QueuedTracker.h, 318
CreateStream
 QueuedCUDATracker, 177
crlb
 SpeedAccResult, 211
CropImage
 testutils.cpp, 274
 testutils.h, 278
CubicBSpline.h
 CUDA_SUPPORTED_FUNC, 287
 ComputeBSplineDerivatives, 287
 ComputeBSplineWeights, 287
 ComputeSplineFitMaxPos, 288
cuda_device
 QTrkSettings, 150
cudaDeviceList
 QueuedCUDATracker.cu, 383
cudaImageList
 alloc, 70
 allocateHostImageBuffer, 70
 bind, 70
 boundaryHit, 70
 capacity, 70
 clear, 70
 computeImagePos, 70
 copyImageToDevice, 71
 copyImageToHost, 71
 copyToDevice, 71
 copyToHost, 71
 count, 74
 data, 74
 emptyList, 72
 free, 72
 fullheight, 72
 fullwidth, 72
 get, 72
 h, 74
 interp, 72
 interpolate, 72
 interpolateFromTexture, 73
 isEmpty, 73
 MaxImageWidth, 69
 numpixels, 73
 pitch, 74
 pixel, 73
 pixel_oobcheck, 73
 pixelAddress, 74
 totalNumBytes, 74
 totalNumPixels, 74
 unbind, 74
 w, 74
 cudaImageList< T >, 68
 cudaImageList
 QueuedCUDATracker.h, 383
 cudaSharedMem
 test.cu, 368
 cudatrack-test/test.cu, 352
 cudatrack/ImageSampler.h, 370
 cudatrack/Kernels.h, 371
 cudatrack/QI.h, 376
 cudatrack/QI_impl.h, 377
 cudatrack/QueuedCUDATracker.cu, 381
 cudatrack/QueuedCUDATracker.h, 383
 cudatrack/cudaImageList.h, 368
 cudatrack/gpu_utils.h, 368
 cudatrack/simplefft.h, 383
d
 LsqSqQuadFit, 116
 pinned_array, 140
d_Qlprofiles
 QI::StreamInstance, 216
d_Qlprofiles_reverse
 QI::StreamInstance, 216
d_com
 QueuedCUDATracker::Stream, 214
d_data
 Image4DMemory::KernelParams, 103
d_imgmeans
 QueuedCUDATracker::Stream, 214
d_locParams
 QueuedCUDATracker::Stream, 214
d_qiparams
 QI::DeviceInstance, 81
d_quadrants
 QI::StreamInstance, 216
d_radialprofiles
 QueuedCUDATracker::Stream, 214
d_radialweights

QI::DeviceInstance, 81
d_resultpos
 QueuedCUDATracker::Stream, 214
d_shiftbuffer
 QI::StreamInstance, 216
d_zlutcmpscores
 QueuedCUDATracker::Stream, 214
DLL_CALLCONV
 dllmacros.h, 289
DLL_EXPORT
 dllmacros.h, 289
data
 Atomic, 38
 cudalmageList, 74
 device_vec, 80
 pinned_array, 138
 QueuedCPUTracker::Job, 101
 TImageData, 227
dataType
 QueuedCPUTracker::Job, 101
dbgCUDAErrorCheck
 gpu_utils.h, 370
DbgCopyResult
 gpu_utils.h, 370
DbgOutputVectorToFile
 gpu_utils.h, 370
dbgPrintResults
 QueuedCPUTracker, 170
dbgout
 utils.cpp, 327
 utils.h, 340
dbgprint
 Matrix3X3, 127
dbgprintf
 utils.cpp, 327
 utils.h, 340
dbgsetlogfile
 utils.cpp, 327
 utils.h, 340
DebugImage
 QueuedTracker, 191
debugImage
 CPUTracker, 66
DeleteAllElems
 utils.h, 340
depth
 Image4DMemory::KernelParams, 103
destroy_tracker
 lv_cputrack_api.cpp, 297
Determinant
 Matrix3X3, 127
Device
 QueuedCUDATracker::Device, 75
device
 QueuedCUDATracker::Stream, 214
device_vec
 ~device_vec, 78
 copyToDevice, 78
 copyToHost, 79
 data, 80
 device_vec, 78
 free, 79
 init, 79
 memsize, 79
 operator std::vector< T >, 80
 operator=, 80
 size, 80
 toVector, 80
 device_vec< T >, 77
DeviceInstance
 QI::DeviceInstance, 81
deviceProp
 QueuedCUDATracker, 187
deviceReport
 QueuedCUDATracker, 187
devices
 QueuedCUDATracker, 187
diag
 Matrix3X3, 127
dimSize
 LVArray, 117
dimSizes
 LVArray2D, 118
 LVArray3D, 119
 LVArrayND, 119
DirExists
 testutils.cpp, 275
 testutils.h, 279
directory
 PathSeperator, 136
distance
 testutils.cpp, 275
 testutils.h, 279
dllmacros.h
 CDLL_EXPORT, 289
 DLL_CALLCONV, 289
 DLL_EXPORT, 289
downsample
 QTrkSettings, 150
downsampleHeight
 QueuedCPUTracker, 170
downsampleWidth
 QueuedCPUTracker, 170
ElectronsPerBit
 Benchmark.cpp, 240
elem
 LVArray, 117
 LVArray2D, 118
 LVArray3D, 119
 LVArrayND, 119
empty
 ImageLUTConfig, 97
emptyKernel
 test.cu, 360
emptyList
 cudalmageList, 72

EnableGainCorrection
 SharedTests.h, 266

EnableRadialZLUTCompareProfile
 QueuedCPUTracker, 158
 QueuedCUDATracker, 178
 QueuedTracker, 191

EnableTextureCache
 QueuedCUDATracker, 178

end
 pinned_array, 138

erf
 utils.h, 340

error
 LocalizationResult, 113

ErrorCluster, 81
 code, 82
 message, 82
 status, 82

Execute
 QI, 141

ExecuteBatch
 QueuedCUDATracker, 178

extension
 PathSeparator, 136

f
 BinaryResultFile, 42
 TextResultFile, 218

FFT2DTracker, 83
 ~FFT2DTracker, 84
 ComputeMax2DInterpolated, 84
 ComputeXCor, 84
 FFT2DTracker, 84
 fft_buf, 84
 fft_buf_mirrored, 84
 GetAutoConvResults, 84
 height, 84
 mirror2D, 84
 plan_bw2D, 84
 plan_fw2D, 84
 width, 84

FFT2D
 BeadFinder.cpp, 282
 CPUTracker::FFT2D, 82

fastjpg.cpp
 FloatToJPEGFile, 289
 ReadJPEGFile, 289
 WriteJPEGFile, 290

FetchResults
 QueuedCPUTracker, 158
 QueuedCUDATracker, 180
 QueuedTracker, 191
 SharedTests.h, 266

fft
 sfft, 35

fft2d
 CPUTracker, 66

fft_backward
 XCor1DBuffer, 234

fft_buf
 FFT2DTracker, 84

fft_buf_mirrored
 FFT2DTracker, 84

fft_forward
 sfft, 35
 XCor1DBuffer, 234

fft_inverse
 sfft, 35

fftPlan
 QI::StreamInstance, 216

File
 outputter::outputModes, 133

file_ext
 utils.cpp, 328
 utils.h, 340

fileError
 ResultManager::FrameCounters, 85

filename
 PathSeparator, 136

Files
 testutils.h, 277

fill_twiddles
 kissfft_utils::traits, 227
 sfft, 36

FillErrorCluster
 LabVIEW datatypes and helper functions, 20

FinalizeLUT
 QueuedCPUTracker, 159
 QueuedCUDATracker, 180
 QueuedTracker, 192

Find
 BeadFinder, 33, 34

firstGuess
 LocalizationResult, 113

FisherElem
 SampleFisherMatrix, 209

FloatToJPEGFile
 fastjpg.cpp, 289
 utils.h, 341

floatToNormalizedInt
 utils.h, 341

Flush
 QueuedCPUTracker, 160
 QueuedCUDATracker, 180
 QueuedTracker, 192
 ResultManager, 199

folder
 outputter, 135

ForceCUDAKernelstoLoad
 Kernels.h, 373

FourierRadialProfile
 CPUTracker, 59

FourierTransform2D
 CPUTracker, 60

frame
 LocalizationJob, 111

FrameCounters

ResultManager::FrameCounters, 85
 frameInfo
 ResultManager::FrameResult, 86
 frameInfoFile
 ResultManager, 204
 frameInfoNames
 ResultManager, 204
 FrameResult
 ResultManager::FrameResult, 86
 frameResults
 ResultManager, 204
 free
 cudaImageList, 72
 device_vec, 79
 Image4DCudaArray, 92
 Image4DMemory, 95
 pinned_array, 138
 TImageData, 225
 freq
 MeasureTime, 130
 fullheight
 cudaImageList, 72
 fullwidth
 cudaImageList, 72
 G2MLE_Compute
 Kernels.h, 373
 gauss2D_iterations
 QTrkSettings, 150
 gauss2D_sigma
 QTrkSettings, 150
 Gauss2DTest
 SharedTests.h, 267
 GaussMask
 testutils.cpp, 275
 testutils.h, 279
 gc_gain
 QueuedCUDATracker, 187
 gc_gainFactor
 QueuedCPUTracker, 170
 QueuedCUDATracker, 187
 gc_mutex
 QueuedCPUTracker, 170
 QueuedCUDATracker, 187
 gc_offset
 QueuedCUDATracker, 187
 gc_offsetFactor
 QueuedCPUTracker, 170
 QueuedCUDATracker, 187
 generate_image_from_lut
 lv_cputrack_api.cpp, 297
 generate_test_image
 lv_cputrack_api.cpp, 297
 GenerateGaussianSpotImage
 utils.cpp, 328
 utils.h, 341
 GenerateImageFromLUT
 utils.cpp, 328
 utils.h, 341
 GenerateLUT
 BenchmarkLUT, 40
 GenerateSample
 BenchmarkLUT, 40
 GenerateTestImage
 QueuedCPUTracker, 160
 utils.cpp, 329
 utils.h, 342
 GenerateZLUTFittingCurve
 main.cpp, 247
 get
 Atomic, 37
 cudaImageList, 72
 LVArray2D, 118
 get_ZLUT
 lv_cputrack_api.cpp, 298
 get_debug_img_as_array
 lv_cputrack_api.cpp, 297
 GetAutoConvResults
 FFT2DTracker, 84
 GetBeadPositions
 ResultManager, 199
 GetBestCUDADevice
 QueuedCUDATracker.cu, 382
 GetCPUCount
 Threads, 223
 GetConfigValues
 QueuedCPUTracker, 160
 QueuedCUDATracker, 180
 QueuedTracker, 192
 GetCurrentOutputPath
 utils.cpp, 329
 utils.h, 343
 GetDebugImage
 CPUTracker, 60
 QueuedCPUTracker, 160
 QueuedTracker, 192
 GetDirectoryFromPath
 utils.cpp, 329
 utils.h, 343
 getExtent
 Image4DCudaArray, 92
 GetFormattedTimeString
 utils.cpp, 330
 utils.h, 343
 GetFrameCount
 ResultManager, 200
 GetFrameCounters
 ResultManager, 200
 GetHeight
 CPUTracker, 60
 GetImageLUTByIndex
 QueuedCPUTracker, 160
 GetImagePos
 Image4DMemory::KernelParams, 103
 getImagePos
 Image4DCudaArray, 92
 Image4DCudaArray::KernelInst, 101

GetImageZLUTSize
 QueuedCPUTracker, 161
 QueuedTracker, 192
GetImageZLUT
 QueuedCPUTracker, 161
 QueuedTracker, 192
getImgAddr
 Image4DMemory, 95
GetLocalModuleFilename
 utils.cpp, 330
 utils.h, 343
GetLocalModulePath
 utils.cpp, 330
 utils.h, 343
GetNewItem
 ThreadPool, 220
GetNextJob
 QueuedCPUTracker, 161
GetOuterEdges
 testutils.cpp, 275
 testutils.h, 279
getPath
 test.cu, 360
GetPixel
 CPUTracker, 60
GetPreciseTime
 utils.cpp, 330
 utils.h, 344
GetProfileReport
 QueuedCPUTracker, 161
 QueuedCUDATracker, 181
 QueuedTracker, 192
GetQueueLength
 QueuedCPUTracker, 161
 QueuedCUDATracker, 181
 QueuedTracker, 192
GetRadialZLUTCompareProfile
 QueuedCPUTracker, 162
 QueuedCUDATracker, 181
 QueuedTracker, 193
GetRadialZLUTSize
 QueuedCPUTracker, 162
 QueuedCUDATracker, 182
 QueuedTracker, 193
GetRadialZLUT
 CPUTracker, 60
 QueuedCPUTracker, 162
 QueuedCUDATracker, 181
 QueuedTracker, 193
 ZLUTParams, 234
GetReadyStream
 QueuedCUDATracker, 182
GetResultCount
 QueuedCPUTracker, 162
 QueuedCUDATracker, 182
 QueuedTracker, 193
GetResults
 ResultManager, 200
getResults
 QueuedCUDATracker::KernelProfileTime, 104
GetTracker
 ResultManager, 200
GetWarnings
 QueuedTracker, 193
GetWidth
 CPUTracker, 60
GetZLUTBiasCorrection
 QueuedTracker, 193
GetZLUTByIndex
 QueuedCPUTracker, 163
gpu_utils.h
 CUBOTH, 369
 CUDA_SUPPORTED_FUNC, 369
 CheckCUDAError, 369
 dbgCUDAErrorCheck, 370
 DbgCopyResult, 370
 DbgOutputVectorToFile, 370
 outputTotalGPUMemUse, 370
h
 cudaImageList, 74
 ImageLUTConfig, 97
 LsqSqQuadFit, 116
 TImageData, 227
 Threads::Mutex, 132
HALF_OF
 kissfft, 106
hasFrameInfo
 ResultManager::FrameResult, 86
height
 CPUTracker, 66
 FFT2DTracker, 84
 QTrkSettings, 151
hostImageBuf
 QueuedCUDATracker::Stream, 214
io
 CPUTracker::Gauss2DResult, 87
imag
 sfft::complex, 46
Image4DCudaArray
 ~Image4DCudaArray, 89
 array, 93
 bind, 90
 clear, 90
 copyImageToDevice, 90
 copyImageToHost, 91
 copyToDevice, 91
 copyToHost, 91
 free, 92
 getExtent, 92
 getImagePos, 92
 Image4DCudaArray, 89
 imgh, 93
 imgw, 93
 kernellInst, 92
 layerh, 93

layerw, 93
 nlayers, 93
 numImg, 93
 unbind, 92
Image4DCudaArray< T >, 88
Image4DCudaArray< T >::KernelInst, 101
Image4DCudaArray::KernelInst
 getImagePos, 101
 imgh, 102
 imgw, 102
 layerw, 102
 readSurfacePixel, 101
 writeSurfacePixel, 102
Image4DMemory
 ~Image4DMemory, 94
 bind, 94
 clear, 94
 copyImageToDevice, 94
 copyImageToHost, 94
 copyToDevice, 95
 copyToHost, 95
 free, 95
 getImgAddr, 95
 Image4DMemory, 94
 kp, 96
 layers, 96
 read, 95
 rows, 96
 totalImg, 96
 unbind, 95
 write, 96
Image4DMemory< T >, 93
Image4DMemory< T >::KernelParams, 102
Image4DMemory::KernelParams
 cols, 103
 d_data, 103
 depth, 103
 GetImagePos, 103
 imgh, 103
 imgw, 103
 pitch, 103
image_lut
 QueuedCPUTracker, 170
image_lut_dims
 QueuedCPUTracker, 170
image_lut_dz
 QueuedCPUTracker, 170
image_lut_dz2
 QueuedCPUTracker, 170
image_lut_nElem_per_bead
 QueuedCPUTracker, 170
image_lut_surface
 Kernels.h, 376
image_sampler_texture_linear
 ImageSampler.h, 371
image_sampler_texture_nearest
 ImageSampler.h, 371
imageBufMutex
 QueuedCUDATracker::Stream, 215
imageCopy
 QueuedCUDATracker::KernelProfileTime, 104
imageCopyDone
 QueuedCUDATracker::Stream, 215
ImageData
 QueuedTracker.h, 317
 utils.h, 337
ImageDataad
 utils.h, 337
ImageLUT_Sample
 Kernels.h, 374
ImageLUTConfig, 96
 empty, 97
 h, 97
 nLUTs, 97
 planes, 97
 w, 97
 xscale, 97
 yscale, 97
ImageLUTHeight
 QueuedCPUTracker, 163
ImageLUTNumBeads
 QueuedCPUTracker, 163
ImageLUTWidth
 QueuedCPUTracker, 163
ImageLUT
 QueuedCUDATracker, 174
imageMean
 LocalizationResult, 113
ImageSampler.h
 image_sampler_texture_linear, 371
 image_sampler_texture_nearest, 371
 ImageSampler_Tex, 371
ImageSampler_InterpolatedTexture, 97
 BindTexture, 97
 Index, 97
 Interpolated, 98
 UnbindTexture, 98
ImageSampler_MemCopy, 98
 BindTexture, 98
 Index, 98
 Interpolated, 98
 UnbindTexture, 99
ImageSampler_SimpleTextureRead, 99
 BindTexture, 99
 Index, 99
 Interpolated, 99
 ofs, 100
 UnbindTexture, 100
ImageSampler_Tex
 ImageSampler.h, 371
Images
 outputter::outputModes, 133
 testutils.h, 277
images
 BaseKernelParams, 38
 QueuedCUDATracker::Stream, 215

imapDone
 QueuedCUDATracker::Stream, 215

img
 ZLUTParams, 235

img_distance
 BeadFinder::Config, 49

img_mean
 Benchmark.cpp, 240

img_sigma
 Benchmark.cpp, 240

ImgDeriv
 SampleFisherMatrix, 209

imgMeans
 QueuedCUDATracker::Stream, 215

imgh
 Image4DCudaArray, 93
 Image4DCudaArray::KernelInst, 102
 Image4DMemory::KernelParams, 103

imgmeans
 BaseKernelParams, 38

imgw
 Image4DCudaArray, 93
 Image4DCudaArray::KernelInst, 102
 Image4DMemory::KernelParams, 103

InDebugMode
 main.cpp, 265

inProgress
 ThreadPool, 222

Index
 ImageSampler_InterpolatedTexture, 97
 ImageSampler_MemCopy, 98
 ImageSampler_SimpleTextureRead, 99

index
 QueuedCUDATracker::Device, 76

Init
 QI, 141

init
 device_vec, 79
 outputter, 134
 pinned_array, 139

InitDevice
 QI, 142

InitStream
 QI, 142

initialPos
 LocalizationJob, 111

InitializeDeviceList
 QueuedCUDATracker, 182

Inter
 main.cpp, 242

interp
 cudalImageList, 72

Interpolate
 utils.h, 344

interpolate
 cudalImageList, 72
 Kernels.h, 374
 TImageData, 225

Interpolate1D
 utils.h, 344

interpolate1D
 TImageData, 225

interpolateFromTexture
 cudalImageList, 73

Interpolated
 ImageSampler_InterpolatedTexture, 98
 ImageSampler_MemCopy, 98
 ImageSampler_SimpleTextureRead, 99

Inverse
 Matrix3X3, 127

InverseTranspose
 Matrix3X3, 128

IsDone
 ThreadPool, 220

isEmpty
 cudalImageList, 73

IsExecutionDone
 QueuedCUDATracker::Stream, 214

IsIdle
 QueuedCPUTracker, 163
 QueuedCUDATracker, 183
 QueuedTracker, 193

isNaN
 utils.h, 344

ItemDone
 ThreadPool, 220

Iterate
 QI, 142

iterations
 QIParams, 144

Job
 QueuedCPUTracker::Job, 100

job
 LocalizationResult, 113
 QueuedCPUTracker::Job, 101

JobCount
 QueuedCUDATracker::Stream, 214

jobCount
 QueuedCPUTracker, 170

JobFinished
 QueuedCPUTracker, 163

jobQueueMutex
 QueuedCUDATracker, 187

jobs
 QueuedCPUTracker, 170
 QueuedCUDATracker::Stream, 215

jobs_buffer
 QueuedCPUTracker, 170

jobs_buffer_mutex
 QueuedCPUTracker, 170

jobs_mutex
 QueuedCPUTracker, 170

jobsInProgress
 QueuedCPUTracker, 170

k

LsqSqQuadFit, 117
 KeepInsideBoundaries
 CPUTracker, 60
 kernelInst
 Image4DCudaArray, 92
 KernelProfileTime
 QueuedCUDATracker::KernelProfileTime, 104
 Kernels.h
 ApplyOffsetGain, 372
 BgCorrectedCOM, 372, 373
 ForceCUDAKernelsToLoad, 373
 G2MLE_Compute, 373
 image_lut_surface, 376
 ImageLUT_Sample, 374
 interpolate, 374
 ZLUT_ComputeProfileMatchScores, 374
 ZLUT_ComputeZ, 375
 ZLUT_NormalizeProfiles, 375
 ZLUT_ProfilesToZLUT, 375
 ZLUT_RadialProfileKernel, 376
 kf_bfly2
 kissfft, 106
 kf_bfly3
 kissfft, 106
 kf_bfly4
 kissfft, 107
 kf_bfly5
 kissfft, 107
 kf_bfly_generic
 kissfft, 108
 kf_work
 kissfft, 109
 kissfft
 -inverse, 110
 -nfft, 110
 -stageRadix, 110
 -stageRemainder, 110
 -twiddles, 110
 C_ADDTO, 105
 C_ADD, 105
 C_FIXDIV, 106
 C_MULBYSCALAR, 106
 C_MUL, 106
 C_SUB, 106
 cpx_type, 105
 HALF_OF, 106
 kf_bfly2, 106
 kf_bfly3, 106
 kf_bfly4, 107
 kf_bfly5, 107
 kf_bfly_generic, 108
 kf_work, 109
 kissfft, 105
 nfft, 109
 S_MUL, 109
 scalar_type, 105
 traits_type, 105
 transform, 109
 kissfft< T_Scalar, T_traits >, 104
 kissfft_utils, 34
 kissfft_utils::traits
 cpx_type, 227
 fill_twiddles, 227
 prepare, 227
 scalar_type, 227
 kissfft_utils::traits< T_scalar >, 227
 kp
 Image4DMemory, 96
 LT_ClearFirstFourPixels
 qtrk_c_api.h, 314
 LT_Force32Bit
 qtrk_c_api.h, 314
 LT_FourierLUT
 qtrk_c_api.h, 314
 LT_Gaussian2D
 qtrk_c_api.h, 314
 LT_LocalizeZWeighted
 qtrk_c_api.h, 314
 LT_LocalizeZ
 qtrk_c_api.h, 314
 LT_NormalizeProfile
 qtrk_c_api.h, 314
 LT_OnlyCOM
 qtrk_c_api.h, 314
 LT_QI
 qtrk_c_api.h, 314
 LT_XCor1D
 qtrk_c_api.h, 314
 LT_ZLUTAlign
 qtrk_c_api.h, 314
 LUTProfMaxQuadraticFit
 CPUTracker, 51
 LUTProfMaxSimpleInterp
 CPUTracker, 51
 LUTProfMaxSplineFit
 CPUTracker, 51
 LUTProfileCompare
 CPUTracker, 61
 LUTProfileCompareAdjustedWeights
 CPUTracker, 63
 LUTProfileMaxComputeMode
 CPUTracker, 51
 LVArray
 dimSize, 117
 elem, 117
 LVArray< T >, 117
 LVArray2D< T >, 117
 LVArray2D
 dimSizes, 118
 elem, 118
 get, 118
 numElem, 118
 xy, 118
 LVArray3D< T >, 118
 LVArray3D
 dimSizes, 119

elem, 119
numElem, 119
LVArrayND< T, N >, 119
LVArrayND
dimSizes, 119
elem, 119
numElem, 119
LVDataType< double >, 120
code, 120
LVDataType< float >, 120
code, 121
LVDataType< int16_t >, 121
code, 121
LVDataType< int32_t >, 121
code, 122
LVDataType< int64_t >, 122
code, 122
LVDataType< int8_t >, 122
code, 123
LVDataType< std::complex< double > >, 123
code, 123
LVDataType< std::complex< float > >, 123
code, 124
LVDataType< T >, 120
LVDataType< uint16_t >, 124
code, 124
LVDataType< uint32_t >, 124
code, 125
LVDataType< uint64_t >, 125
code, 125
LVDataType< uint8_t >, 125
code, 126
LVGetStringArray
LabVIEW datatypes and helper functions, 20
LabVIEW datatypes and helper functions, 19
ArgumentErrorMsg, 20
FillErrorCluster, 20
LVGetStringArray, 20
ppFloatArray, 20
ppFloatArray2, 20
ResizeLVArray, 21
ResizeLVArray2D, 21
ResizeLVArray3D, 21
SetLVString, 21
ValidateTracker, 22
labview.h
_STDINT_H, 294
lastSaveFrame
ResultManager::FrameCounters, 85
layerh
Image4DCudaArray, 93
layers
Image4DMemory, 96
layerw
Image4DCudaArray, 93
Image4DCudaArray::KernelInst, 102
length
vector3, 230
Lerp
utils.h, 344
linspace
SharedTests.h, 268
listDevices
test.cu, 360
Load
BenchmarkLUT, 40, 41
LoadRow
BinaryResultFile, 42
ResultFile, 197
TextResultFile, 217
LocMode_t
qtrk_c_api.h, 314
locParams
BaseKernelParams, 38
QueuedCUDATracker::Stream, 215
localizationDone
QueuedCUDATracker::Stream, 215
LocalizationJob, 110
frame, 111
initialPos, 111
LocalizationJob, 111
timestamp, 111
zlutIndex, 111
LocalizationParams, 112
zlutIndex, 112
zlutPlane, 112
LocalizationResult, 112
error, 113
firstGuess, 113
imageMean, 113
job, 113
pos, 113
pos2D, 113
localizationsDone
ResultManager::FrameCounters, 85
localizeFlags
QueuedCUDATracker::Stream, 215
localizeMode
QueuedCPUTracker, 170
QueuedCUDATracker, 187
LocalizeModeEnum
qtrk_c_api.h, 314
lock
QueuedCPUTracker::Thread, 218
Threads::Mutex, 131
lockCount
Threads::Mutex, 132
logFilename
utils.cpp, 335
logspace
SharedTests.h, 268
lostFrames
ResultManager::FrameCounters, 85
LsqQuadraticFit.h
CUDA_SUPPORTED_FUNC, 294
LsqSqQuadFit

a, 116
 b, 116
 c, 116
 calculate, 115
 compute, 115
 computeDeriv, 115
 computeSums, 115
 d, 116
 h, 116
 k, 117
 LsqSqQuadFit, 114
 maxPos, 116
 vertexForm, 116
 xoffset, 117
 LsqSqQuadFit< T >, 114
 LsqSqQuadFit< T >::Coeff, 44
 LsqSqQuadFit::Coeff
 abc, 45
 s00, 45
 s01, 45
 s10, 45
 s11, 45
 s20, 45
 s21, 45
 s30, 45
 s40, 45
 lut_h
 BenchmarkLUT, 41
 lut_w
 BenchmarkLUT, 41
 lv_cputrack_api.cpp
 compute_asymmetry, 295
 compute_com, 295
 compute_crp, 295
 compute_qi, 296
 compute_radial_profile, 296
 compute_xcor, 296
 compute_z, 296
 create_tracker, 297
 destroy_tracker, 297
 generate_image_from_lut, 297
 generate_test_image, 297
 get_ZLUT, 298
 get_debug_img_as_array, 297
 set_ZLUT, 299
 set_image_float, 298
 set_image_from_memory, 298
 set_image_u16, 298
 set_image_u8, 298
 lv_queuetrk_api.cpp
 CheckImageInput, 302
 qtrk_build_lut_plane, 302
 qtrk_compute_fisher, 303
 qtrk_compute_zlut_bias_table, 303
 qtrk_enable_zlut_cmpprof, 303
 qtrk_finalize_lut, 304
 qtrk_find_beads, 304
 qtrk_free_all, 304
 qtrk_generate_gaussian_spot, 304
 qtrk_get_computed_config, 304
 qtrk_get_debug_image, 304
 qtrk_get_image_lut, 305
 qtrk_get_queue_len, 305
 qtrk_get_zlut_cmpprof, 305
 qtrk_set_image_lut, 305
 qtrk_set_localization_mode, 305
 qtrk_set_logfile_path, 306
 qtrk_set_pixel_calib, 306
 qtrk_set_pixel_calib_factors, 306
 qtrk_set_zlut_bias_table, 306
 qtrk_simulate_tracking, 307
 qtrkcuda_get_device, 307
 test_array_passing, 308
 trackerList, 308
 trackerListMutex, 308
 lv_resultmanager_api.cpp
 rm_create, 309
 rm_destroy, 309
 rm_destroy_all, 309
 rm_flush, 309
 rm_getbeadresults, 309
 rm_getconfig, 310
 rm_getframecounters, 310
 rm_getresults, 310
 rm_instances, 311
 rm_removebead, 310
 rm_set_tracker, 310
 rm_store_frame_info, 311
 ValidRM, 311
 m
 Atomic, 38
 Matrix3X3, 129
 MARKPIXELI
 cpu_tracker.cpp, 285
 MARKPIXEL
 cpu_tracker.cpp, 285
 MIN_RADPROFILE_SMP_COUNT
 QueuedTracker.h, 317
 main
 main.cpp, 247
 test.cu, 360
 main.cpp
 ANGSTEPF, 265
 AccBiasTest, 242
 AutoBeadFindTest, 243
 Backg, 242
 BenchmarkParams, 243
 BuildConvergenceMap, 244
 BuildZLUT, 245
 CRP_TestGeneratedData, 247
 ComputeRadialWeights, 246
 CorrectedRadialProfileTest, 246
 GenerateZLUTFittingCurve, 247
 InDebugMode, 265
 Inter, 242
 main, 247

ManTest, 248
OnePixelTest, 250
OutputProfileImg, 250
PixelationErrorTest, 251
PrintMenu, 251
ROIDis, 242
RWDerivative, 242
RWNone, 242
RWRadial, 242
RWStetson, 242
RWUniform, 242
RWeightMode, 242
RescaleLUT, 251
RunCOMAndQI, 251
RunTest, 252
RunZTrace, 252
ScatterBiasArea, 255
SelectTests, 255
SimpleTest, 256
Skew, 242
SkewParam, 257
SmallImageTest, 257
SpeedTest, 257
TestBSplineMax, 260
TestBackground, 259
TestBias, 259
TestBoundCheck, 260
TestFourierLUTOnDataset, 261
TestFourierLUT, 260
TestInterference, 261
TestQuadrantAlign, 261
TestROIDisplacement, 262
TestSkew, 262
TestZLUTAlign, 262
TestZRange, 263
TestZRangeBias, 265
Tests, 242
WriteRadialProf, 265
major
 CUDADeviceInfo, 68
ManTest
 main.cpp, 248
manager
 QueuedCPUTracker::Thread, 219
Matrix3X3, 126
 at, 127
 dbgprint, 127
 Determinant, 127
 diag, 127
 Inverse, 127
 InverseTranspose, 128
 m, 129
 Matrix3X3, 127
 operator*!=, 128
 operator(), 128
 operator+=, 128
 operator[], 129
 row, 129
max_
 test, 129
max_a
 ComputeMaxInterp, 48
max_a
 BenchmarkLUT, 41
max_b
 BenchmarkLUT, 41
max_c
 BenchmarkLUT, 41
maxFramesInMemory
 ResultManagerConfig, 205
MaxImageWidth
 cudaImageList, 69
maxPos
 LsqSqQuadFit, 116
maxQueueSize
 QueuedCPUTracker, 170
maxRadius
 QIParams, 144
 ZLUTParams, 235
maxValue
 SampleFisherMatrix, 209
mean
 CPUTracker, 66
 TImageData, 226
meanErr
 RunTrackerResults, 207
MeanStDevError
 SharedTests.h, 268
MeasureTime, 129
 ~MeasureTime, 130
 freq, 130
 MeasureTime, 130
 name, 130
 time, 130
memsize
 device_vec, 79
 pinned_array, 139
 QI::StreamInstance, 216
message
 ErrorCluster, 82
min_
 ComputeMaxInterp, 49
MinPixelDistance
 BeadFinder::Config, 49
minRadius
 QIParams, 144
 ZLUTParams, 235
minor
 CUDADeviceInfo, 68
mirror2D
 FFT2DTracker, 84
modes
 outputter, 135
msg
 Threads::Mutex, 131
mul_conjugate
 test.cu, 361

multiProcCount
 CUDADeviceInfo, 68
 Mutex
 Threads::Mutex, 131
 mutex
 QueuedCPUTracker::Thread, 219
 my_error_mgr, 132
 pub, 132

 n
 pinned_array, 140
 nLUTs
 ImageLUTConfig, 97
 name
 CUDADeviceInfo, 68
 MeasureTime, 130
 Threads::Mutex, 132
 NearestPowerOf2
 utils.cpp, 330
 utils.h, 345
 NearestPowerOf3
 utils.cpp, 331
 utils.h, 345
 NearestPowerOfTwo
 test.cu, 361
 newFile
 outputter, 134
 NextPowerOf2
 BeadFinder.cpp, 282
 nfft
 kissfft, 109
 njobs
 BaseKernelParams, 38
 nlayers
 Image4DCudaArray, 93
 Normalize
 CPUTracker, 63
 normalize
 TImageData, 226
 utils.h, 345
 NormalizeRadialProfile
 utils.cpp, 331
 utils.h, 345
 NormalizeZLUT
 utils.cpp, 331
 utils.h, 345
 normprof
 BenchmarkLUT, 41
 numBeads
 ResultManagerConfig, 205
 numElem
 LVArray2D, 118
 LVArray3D, 119
 LVArrayND, 119
 NumFilesInDir
 testutils.cpp, 275
 testutils.h, 280
 numFrameInfoColumns
 ResultManagerConfig, 205

 numImg
 Image4DCudaArray, 93
 NumJpgInDir
 testutils.cpp, 276
 testutils.h, 280
 numPixels
 TImageData, 226
 NumThreads
 QueuedCPUTracker, 163
 numThreads
 QTrkSettings, 151
 QI, 144
 QueuedCUDATracker, 187
 numpixels
 cudalImageList, 73

 offset
 ResultManagerConfig, 205
 ofs
 ImageSampler_SimpleTextureRead, 100
 OnePixelTest
 main.cpp, 250
 operator std::vector< T >
 device_vec, 80
 operator T
 Atomic, 37
 operator*
 sfft::complex, 47
 vector3, 230, 232
 operator*=
 Matrix3X3, 128
 sfft::complex, 47
 vector3, 231
 operator()
 Matrix3X3, 128
 operator+
 sfft::complex, 47
 vector3, 231
 operator+=
 Matrix3X3, 128
 sfft::complex, 47
 vector3, 231
 operator-
 sfft::complex, 47
 vector3, 231
 operator-=
 vector3, 232
 operator/
 vector3, 232
 operator/=
 vector3, 232
 operator=
 Atomic, 38
 CImageData, 44
 device_vec, 80
 pinned_array, 139
 operator[]
 Matrix3X3, 129
 pinned_array, 139

TImageData, 226
output
RunTrackerResults, 207
outputArray
 outputter, 134
outputFile
 outputter, 135
 ResultManager, 204
outputImage
 outputter, 135
OutputMemoryUse
 QueuedCUDATracker::Stream, 214
OutputModes
 testutils.h, 277
OutputProfileImg
 main.cpp, 250
outputString
 outputter, 135
outputTotalGPUMemUse
 gpu_utils.h, 370
outputter, 133
 ~outputter, 134
 folder, 135
 init, 134
 modes, 135
 newFile, 134
 outputArray, 134
 outputFile, 135
 outputImage, 135
 outputString, 135
 outputter, 134
outputter::outputModes, 132
 Console, 133
 File, 133
 Images, 133

PDT_BytesPerPixel
 QueuedTracker.h, 318
parallel_for
 threads.h, 323
param
 Threads::Handle, 87
params
 QI, 144
PathSeperator, 135
 directory, 136
 extension, 136
 filename, 136
 PathSeperator, 136
pinned_array
 ~pinned_array, 137
 begin, 138
 d, 140
 data, 138
 end, 138
 free, 138
 init, 139
 memsize, 139
 n, 140

operator=, 139
operator[], 139
pinned_array, 137, 138
 size, 139
pinned_array< T, flags >, 136
pitch
 cudaImageList, 74
 Image4DMemory::KernelParams, 103
 TImageData, 226
pixel
 cudaImageList, 73
pixel_oobcheck
 cudaImageList, 73
pixel_t
 cpu_tracker.h, 287
pixelAddress
 cudaImageList, 74
PixelationErrorTest
 main.cpp, 251
plan_bw2D
 FFT2DTracker, 84
plan_fw2D
 FFT2DTracker, 84
planes
 ImageLUTConfig, 97
 ZLUTParams, 235
pos
 CPUTracker::Gauss2DResult, 87
 LocalizationResult, 113
pos2D
 LocalizationResult, 113
Position
 BeadFinder::Position, 140
ppFloatArray
 LabVIEW datatypes and helper functions, 20
ppFloatArray2
 LabVIEW datatypes and helper functions, 20
prepare
 kissfft_utils::traits, 227
PrintMenu
 main.cpp, 251
ProcessArray
 ThreadPool, 221
ProcessJob
 QueuedCPUTracker, 164
processJobs
 QueuedCPUTracker, 170
processedFrames
 ResultManager::FrameCounters, 85
ProfileSpeedVsROI
 test.cu, 361
pub
 my_error_mgr, 132
QFF_Force32Bit
 API - LabVIEW, 24
QI.h
 qicomplex_t, 377
 qivalue_t, 377

QI::DeviceInstance, 81
 ~DeviceInstance, 81
 d_qiparams, 81
 d_radialweights, 81
 DeviceInstance, 81
 qi_trigtable, 81
 QI::StreamInstance, 215
 ~StreamInstance, 216
 d_Qlprofiles, 216
 d_Qlprofiles_reverse, 216
 d_quadrants, 216
 d_shiftbuffer, 216
 fftPlan, 216
 memsize, 216
 stream, 216
 QI_ComputeAxisOffset
 QI_impl.h, 378
 QI_ComputeOffset
 CPUTracker, 63
 QI_ComputeProfile
 QI_impl.h, 378
 QI_ComputeQuadrants
 QI_impl.h, 379
 QI_LSQFIT_NWEIGHTS
 QueuedTracker.h, 317
 QI_LSQFIT_WEIGHTS
 QueuedTracker.h, 317
 QI_MultiplyWithConjugate
 QI_impl.h, 379
 QI_OffsetPositions
 QI_impl.h, 380
 QI_QuadrantsToProfiles
 QI_impl.h, 380
 QI_impl.h
 ComputeQuadrantProfile, 377
 QI_ComputeAxisOffset, 378
 QI_ComputeProfile, 378
 QI_ComputeQuadrants, 379
 QI_MultiplyWithConjugate, 379
 QI_OffsetPositions, 380
 QI_QuadrantsToProfiles, 380
 QICompare
 test.cu, 362
 QIParams, 144
 cos_sin_table, 144
 iterations, 144
 maxRadius, 144
 minRadius, 144
 radialSteps, 144
 trigtablesize, 144
 QIWeights
 cpu_tracker.cpp, 286
 QTRK_PixelDataType
 qtrk_c_api.h, 314
 QTrkBuidLUT
 API - C, 29
 QTrkCUDA_UseAll
 qtrk_c_api.h, 314
 QTrkCUDA_UseBest
 qtrk_c_api.h, 314
 QTrkCUDA_UseList
 qtrk_c_api.h, 314
 QTrkClearResults
 API - C, 29
 QTrkCompareTest
 test.cu, 362
 QTrkComputedConfig, 145
 QTrkComputedConfig, 146
 qi_angstepspq, 147
 qi_maxradius, 147
 qi_radialsteps, 147
 Update, 146
 WriteToFile, 146
 WriteToLog, 146
 zlut-angularsteps, 147
 zlut_maxradius, 147
 zlut_radialsteps, 147
 QTrkCreateInstance
 API - C, 30
 QTrkFetchResults
 API - C, 30
 QTrkFinalizeLUT
 API - C, 30
 QTrkFloat
 qtrk_c_api.h, 315
 QTrkFlush
 API - C, 30
 QTrkFreeInstance
 API - C, 30
 QTrkGetComputedConfig
 API - C, 30
 QTrkGetProfileReport
 API - C, 31
 QTrkGetQueueLength
 API - C, 31
 QTrkGetRadialZLUTSize
 API - C, 31
 QTrkGetRadialZLUT
 API - C, 31
 QTrkGetResultCount
 API - C, 31
 QTrkGetWarnings
 API - C, 31
 QTrkIsIdle
 API - C, 31
 QTrkScheduleFrame
 API - C, 32
 QTrkScheduleLocalization
 API - C, 32
 QTrkSetLocalizationMode
 API - C, 32
 QTrkSetRadialZLUT
 API - C, 32
 QTrkSettings, 148
 com_bgcorrection, 150
 cuda_device, 150

downsample, 150
gauss2D_iterations, 150
gauss2D_sigma, 150
height, 151
numThreads, 151
QTrkSettings, 150
qi_angstep_factor, 151
qi_angular_coverage, 151
qi_iterations, 151
qi_minradius, 151
qi_radial_coverage, 151
qi_roi_coverage, 151
testRun, 151
width, 152
xc1_iterations, 152
xc1_profileLength, 152
xc1_profileWidth, 152
zlut_angular_coverage, 152
zlut_minradius, 152
zlut_radial_coverage, 152
zlut_roi_coverage, 152

QTrkU16
 qtrk_c_api.h, 315

QTrkU8
 qtrk_c_api.h, 315

qa_fft_backward
 CPUTracker, 67

qa_fft_forward
 CPUTracker, 67

qalign
 QueuedCUDATracker, 187

qalign_instance
 QueuedCUDATracker::Device, 77
 QueuedCUDATracker::Stream, 215

qalignDone
 QueuedCUDATracker::Stream, 215

QI, 140
 batchSize, 144
 blocks, 141
 Execute, 141
 Init, 141
 InitDevice, 142
 InitStream, 142
 Iterate, 142
 numThreads, 144
 params, 144
 qi_FFT_length, 144
 threads, 143

qi
 QueuedCUDATracker, 187
 QueuedCUDATracker::KernelProfileTime, 104

qi_FFT_length
 QI, 144

qi_angstep_factor
 QTrkSettings, 151

qi_angstepspq
 QTrkComputedConfig, 147

qi_angular_coverage
 QTrkSettings, 151
 qi_fft_backward
 CPUTracker, 67
 qi_fft_forward
 CPUTracker, 67
 qi_instance
 QueuedCUDATracker::Device, 77
 QueuedCUDATracker::Stream, 215
 qi_iterations
 QTrkSettings, 151
 qi_maxradius
 QTrkComputedConfig, 147
 qi_minradius
 QTrkSettings, 151
 qi_radial_coverage
 QTrkSettings, 151
 qi_radialbinweights
 QueuedCPUTracker, 170
 qi_radialsteps
 CPUTracker, 67
 QTrkComputedConfig, 147
 qi_roi_coverage
 QTrkSettings, 151
 qi_trigtable
 QI::DeviceInstance, 81
 qiDone
 QueuedCUDATracker::Stream, 215
 qicomplex_t
 QI.h, 377
 qivalue_t
 QI.h, 377
 qselect
 utils.h, 346
 qtrk, 34
 ResultManager, 204
 qtrk::hash_map< TKey, T >, 88
 qtrk::hash_set< T >, 88
 qtrk_build_lut_plane
 lv_queuetrk_api.cpp, 302
 qtrk_c_api.h
 LT_ClearFirstFourPixels, 314
 LT_Force32Bit, 314
 LT_FourierLUT, 314
 LT_Gaussian2D, 314
 LT_LocalizeZWeighted, 314
 LT_LocalizeZ, 314
 LT_NormalizeProfile, 314
 LT_OnlyCOM, 314
 LT_QI, 314
 LT_XCor1D, 314
 LT_ZLUTAlign, 314
 LocMode_t, 314
 LocalizeModeEnum, 314
 QTRK_PixelDataType, 314
 QTrkCUDA_UseAll, 314
 QTrkCUDA_UseBest, 314
 QTrkCUDA_UseList, 314
 QTrkFloat, 315

QTrkU16, 315
 QTrkU8, 315
 qtrk_clear_results
 API - LabVIEW, 24
 qtrk_compute_fisher
 lv_queuetrk_api.cpp, 303
 qtrk_compute_zlut_bias_table
 lv_queuetrk_api.cpp, 303
 qtrk_create
 API - LabVIEW, 24
 qtrk_destroy
 API - LabVIEW, 24
 qtrk_dump_memleaks
 API - LabVIEW, 24
 qtrk_enable_zlut_cmpprof
 lv_queuetrk_api.cpp, 303
 qtrk_finalize_lut
 lv_queuetrk_api.cpp, 304
 qtrk_find_beads
 lv_queuetrk_api.cpp, 304
 qtrk_flush
 API - LabVIEW, 25
 qtrk_free_all
 lv_queuetrk_api.cpp, 304
 qtrk_generate_gaussian_spot
 lv_queuetrk_api.cpp, 304
 qtrk_generate_image_from_lut
 API - LabVIEW, 25
 qtrk_get_ZLUT
 API - LabVIEW, 25
 qtrk_get_computed_config
 lv_queuetrk_api.cpp, 304
 qtrk_get_debug_image
 lv_queuetrk_api.cpp, 304
 qtrk_get_image_lut
 lv_queuetrk_api.cpp, 305
 qtrk_get_profile_report
 API - LabVIEW, 25
 qtrk_get_queue_len
 lv_queuetrk_api.cpp, 305
 qtrk_get_results
 API - LabVIEW, 25
 qtrk_get_zlut_cmpprof
 lv_queuetrk_api.cpp, 305
 qtrk_idle
 API - LabVIEW, 26
 qtrk_queue_array
 API - LabVIEW, 26
 qtrk_queue_float
 API - LabVIEW, 26
 qtrk_queue_frame
 API - LabVIEW, 26
 qtrk_queue_pitchedmem
 API - LabVIEW, 26
 qtrk_queue_u16
 API - LabVIEW, 27
 qtrk_queue_u8
 API - LabVIEW, 27

qtrk_read_timestamp
 API - LabVIEW, 27
 qtrk_resultcount
 API - LabVIEW, 27
 qtrk_set_ZLUT
 API - LabVIEW, 27
 qtrk_set_image_lut
 lv_queuetrk_api.cpp, 305
 qtrk_set_localization_mode
 lv_queuetrk_api.cpp, 305
 qtrk_set_logfile_path
 lv_queuetrk_api.cpp, 306
 qtrk_set_pixel_calib
 lv_queuetrk_api.cpp, 306
 qtrk_set_pixel_calib_factors
 lv_queuetrk_api.cpp, 306
 qtrk_set_zlut_bias_table
 lv_queuetrk_api.cpp, 306
 qtrk_simulate_tracking
 lv_queuetrk_api.cpp, 307
 qtrkcuda_device_count
 API - LabVIEW, 28
 qtrkcuda_get_device
 lv_queuetrk_api.cpp, 307
 qtrkcuda_set_device_list
 API - LabVIEW, 28
 QuadrantAlign
 CPUTracker, 64
 QuadrantAlign_ComputeOffset
 CPUTracker, 65
 quadrantDirs
 CPUTracker, 67
 QueueFrameFlags
 API - LabVIEW, 24
 QueuedCPUTracker, 153
 ~QueuedCPUTracker, 156
 AddJob, 156
 AllocateJob, 156
 ApplyOffsetGain, 156
 BeginLUT, 157
 Break, 157
 BuildLUT, 157
 calib_gain, 170
 calib_offset, 170
 ClearResults, 158
 dbgPrintResults, 170
 downsampleHeight, 170
 downsampleWidth, 170
 EnableRadialZLUTCompareProfile, 158
 FetchResults, 158
 FinalizeLUT, 159
 Flush, 160
 gc_gainFactor, 170
 gc_mutex, 170
 gc_offsetFactor, 170
 GenerateTestImage, 160
 GetConfigValues, 160
 GetDebugImage, 160

GetImageLUTByIndex, 160
GetImageZLUTSize, 161
GetImageZLUT, 161
GetNextJob, 161
GetProfileReport, 161
GetQueueLength, 161
GetRadialZLUTCompareProfile, 162
GetRadialZLUTSize, 162
GetRadialZLUT, 162
GetResultCount, 162
GetZLUTByIndex, 163
image_lut, 170
image_lut_dims, 170
image_lut_dz, 170
image_lut_dz2, 170
image_lut_nElem_per_bead, 170
ImageLUTHeight, 163
ImageLUTNumBeads, 163
ImageLUTWidth, 163
IsIdle, 163
jobCount, 170
JobFinished, 163
jobs, 170
jobs_buffer, 170
jobs_buffer_mutex, 170
jobs_mutex, 170
jobsInProgress, 170
localizeMode, 170
maxQueueSize, 170
NumThreads, 163
ProcessJob, 164
processJobs, 170
qi_radialbinweights, 170
QueuedCPUTracker, 155
quitWork, 171
resultCount, 171
results, 171
results_mutex, 171
ScheduleLocalization, 165
SetConfigValue, 166
SetImageZLUT, 166
SetLocalizationMode, 166
SetPixelCalibrationFactors, 166
SetPixelCalibrationImages, 167
SetRadialWeights, 167
SetRadialZLUT, 168
SetTrackerImage, 168
Start, 168
threads, 171
UpdateZLUTs, 168
WorkerThreadMain, 169
ZLUTSelfTest, 169
zcmp, 171
zlut_buildflags, 171
zlut_cmpprofiles, 171
zlut_count, 171
zlut_enablecmpprof, 171
zlut_planes, 171
zluts, 171
QueuedCPUTracker.cpp
CreateQueuedTracker, 315
SetCUDADevices, 315
QueuedCPUTracker::Job, 100
~Job, 100
data, 101
dataType, 101
Job, 100
job, 101
QueuedCPUTracker::Thread, 218
lock, 218
manager, 219
mutex, 219
Thread, 218
thread, 219
tracker, 219
unlock, 218
QueuedCUDATracker, 171
~QueuedCUDATracker, 175
batchSize, 187
batchesDone, 187
BeginLUT, 175
blocks, 175
BuildLUT, 175
CPU_ApplyOffsetGain, 177
ClearResults, 176
CopyStreamResults, 176
cpu_time, 187
CreateStream, 177
deviceProp, 187
deviceReport, 187
devices, 187
EnableRadialZLUTCompareProfile, 178
EnableTextureCache, 178
ExecuteBatch, 178
FetchResults, 180
FinalizeLUT, 180
Flush, 180
gc_gain, 187
gc_gainFactor, 187
gc_mutex, 187
gc_offset, 187
gc_offsetFactor, 187
GetConfigValues, 180
GetProfileReport, 181
GetQueueLength, 181
GetRadialZLUTCompareProfile, 181
GetRadialZLUTSize, 182
GetRadialZLUT, 181
GetReadyStream, 182
GetResultCount, 182
ImageLUT, 174
InitializeDeviceList, 182
IsIdle, 183
jobQueueMutex, 187
localizeMode, 187
numThreads, 187

qalign, 187
 qi, 187
 QueuedCUDATracker, 174
 quitScheduler, 187
 resultCount, 187
 resultMutex, 187
 results, 188
 ScheduleLocalization, 183
 schedulingThread, 188
 SchedulingThreadEntryPoint, 184
 SchedulingThreadMain, 184
 SetConfigValue, 184
 SetLocalizationMode, 185
 SetPixelCalibrationFactors, 185
 SetPixelCalibrationImages, 185
 SetRadialWeights, 186
 SetRadialZLUT, 186
 StreamUpdateZLUTSize, 186
 streams, 188
 threads, 186
 time, 188
 useTextureCache, 188
 zlut_build_flags, 188
 QueuedCUDATracker.cu
 AddProfilesToZLUT, 382
 checksum, 382
 CreateQueuedTracker, 382
 cudaDeviceList, 383
 GetBestCUDADevice, 382
 SetCUDADevices, 382
 TRK_PROFILE, 382
 QueuedCUDATracker.h
 cudaImageListf, 383
 QueuedCUDATracker::Device, 75
 ~Device, 75
 calib_gain, 76
 calib_offset, 76
 Device, 75
 index, 76
 qalign_instance, 77
 qi_instance, 77
 radial_zlut, 77
 SetPixelCalibrationImages, 76
 SetRadialWeights, 76
 SetRadialZLUT, 76
 zcompareWindow, 77
 zlut_trigtable, 77
 QueuedCUDATracker::KernelProfileTime, 103
 com, 104
 getResults, 104
 imageCopy, 104
 KernelProfileTime, 104
 qi, 104
 zcompute, 104
 zlutAlign, 104
 QueuedCUDATracker::Stream, 212
 ~Stream, 213
 batchStart, 214
 com, 214
 comDone, 214
 d_com, 214
 d_imgmeans, 214
 d_locParams, 214
 d_radialprofiles, 214
 d_resultpos, 214
 d_zlutcmpscores, 214
 device, 214
 hostImageBuf, 214
 imageBufMutex, 215
 imageCopyDone, 215
 images, 215
 imapDone, 215
 imgMeans, 215
 IsExecutionDone, 214
 JobCount, 214
 jobs, 215
 locParams, 215
 localizationDone, 215
 localizeFlags, 215
 OutputMemoryUse, 214
 qalign_instance, 215
 qalignDone, 215
 qi_instance, 215
 qiDone, 215
 results, 215
 State, 213
 state, 215
 Stream, 213
 stream, 215
 StreamExecuting, 213
 StreamIdle, 213
 StreamPendingExec, 213
 zcomputeDone, 215
 QueuedTracker, 188
 ~QueuedTracker, 190
 BeginLUT, 190
 BuildLUT, 190
 cfg, 196
 ClearResults, 190
 ComputeZBiasCorrection, 190
 ConfigValueMap, 190
 DebugImage, 191
 EnableRadialZLUTCompareProfile, 191
 FetchResults, 191
 FinalizeLUT, 192
 Flush, 192
 GetConfigValues, 192
 GetDebugImage, 192
 GetImageZLUTSize, 192
 GetImageZLUT, 192
 GetProfileReport, 192
 GetQueueLength, 192
 GetRadialZLUTCompareProfile, 193
 GetRadialZLUTSize, 193
 GetRadialZLUT, 193
 GetResultCount, 193

GetWarnings, 193
GetZLUTBiasCorrection, 193
IsIdle, 193
QueuedTracker, 190
ScheduleFrame, 193
ScheduleImageData, 194
ScheduleLocalization, 194
SetConfigValue, 194
SetImageZLUT, 195
SetLocalizationMode, 195
SetPixelCalibrationFactors, 195
SetPixelCalibrationImages, 195
SetRadialWeights, 195
SetRadialZLUT, 195
SetZLUTBiasCorrection, 195
ZLUTBiasCorrection, 195
zlut_bias_correction, 196
QueuedTracker.cpp
 WRITEVAR, 316
QueuedTracker.h
 BUILDLUT_BIASCORRECT, 317
 BUILDLUT_FOURIER, 317
 BUILDLUT_IMAGELUT, 317
 BUILDLUT_NORMALIZE, 317
 CopyImageToFloat, 317
 CreateQueuedTracker, 318
 ImageData, 317
 MIN_RADPROFILE_SMP_COUNT, 317
 PDT_BytesPerPixel, 318
 QI_LSQFIT_NWEIGHTS, 317
 QI_LSQFIT_WEIGHTS, 317
 SetCUDADevices, 318
 ZLUT_LSQFIT_NWEIGHTS, 317
 ZLUT_LSQFIT_WEIGHTS, 317
Quit
 ThreadPool, 221
quit
 ResultManager, 204
 ThreadPool, 222
quitScheduler
 QueuedCUDATracker, 187
quitWork
 QueuedCPUTracker, 171
ROIDis
 main.cpp, 242
ROIPosition, 205
 x, 206
 y, 206
RWDerivative
 main.cpp, 242
RWNone
 main.cpp, 242
RWRadial
 main.cpp, 242
RWStetson
 main.cpp, 242
RWUniform
 main.cpp, 242
RWeightMode
 main.cpp, 242
radial_zlut
 QueuedCUDATracker::Device, 77
radialDirs
 CPUTracker, 67
radialSteps
 QIParams, 144
 ZLUTParams, 234
rand_normal
 random_distr.h, 318
rand_poisson
 random_distr.h, 318
rand_uniform
 random_distr.h, 319
random
 vector2, 229
random_distr.h
 rand_normal, 318
 rand_poisson, 318
 rand_uniform, 319
RandomFill
 SharedTests.h, 268
read
 Image4DMemory, 95
ReadCSV
 utils.cpp, 331
 utils.h, 346
ReadJPEGFile
 fastjpg.cpp, 289
 utils.cpp, 332
 utils.h, 346, 347
ReadLUTFile
 utils.cpp, 332
 utils.h, 348
readSurfacePixel
 Image4DCudaArray::KernellInst, 101
ReadToByteBuffer
 utils.cpp, 332
 utils.h, 348
ReadVector3CSV
 utils.cpp, 333
 utils.h, 348
real
 sfft::complex, 47
RecenterAndFilter
 BeadFinder.cpp, 282
RemoveBeadResults
 ResultManager, 200
ResampleLUT
 SharedTests.h, 269
RescaleLUT
 main.cpp, 251
ResizeImage
 testutils.cpp, 276
 testutils.h, 280
ResizeLUT
 SharedTests.h, 269

ResizeLVArray
 LabVIEW datatypes and helper functions, 21
 ResizeLVArray2D
 LabVIEW datatypes and helper functions, 21
 ResizeLVArray3D
 LabVIEW datatypes and helper functions, 21
 resultCount
 QueuedCPUTracker, 171
 QueuedCUDATracker, 187
 ResultFile, 196
 ~ResultFile, 197
 LoadRow, 197
 ResultFile, 197
 SaveRow, 197
 resultFile
 ResultManager, 204
 ResultManager, 197
 ~ResultManager, 198
 CheckResultSpace, 199
 cnt, 204
 Config, 199
 config, 204
 Flush, 199
 frameInfoFile, 204
 frameInfoNames, 204
 frameResults, 204
 GetBeadPositions, 199
 GetFrameCount, 200
 GetFrameCounters, 200
 GetResults, 200
 GetTracker, 200
 outputFile, 204
 qtrk, 204
 quit, 204
 RemoveBeadResults, 200
 resultFile, 204
 ResultManager, 198
 resultMutex, 204
 SaveSection, 201
 SetTracker, 201
 StoreFrameInfo, 201
 StoreResult, 201
 thread, 204
 ThreadLoop, 202
 trackerMutex, 204
 Update, 202
 Write, 202
 WriteBinaryFileHeader, 203
 WriteBinaryResults, 203
 WriteTextResults, 204
 ResultManager.cpp
 BINFILE_VERSION, 319
 ResultManager::FrameCounters, 85
 capturedFrames, 85
 fileError, 85
 FrameCounters, 85
 lastSaveFrame, 85
 localizationsDone, 85
 lostFrames, 85
 processedFrames, 85
 startFrame, 85
 ResultManager::FrameResult, 86
 count, 86
 frameInfo, 86
 FrameResult, 86
 hasFrameInfo, 86
 results, 86
 timestamp, 86
 ResultManagerConfig, 205
 binaryOutput, 205
 maxFramesInMemory, 205
 numBeads, 205
 numFrameInfoColumns, 205
 offset, 205
 scaling, 205
 writeInterval, 205
 resultMutex
 QueuedCUDATracker, 187
 ResultManager, 204
 results
 QueuedCPUTracker, 171
 QueuedCUDATracker, 188
 QueuedCUDATracker::Stream, 215
 ResultManager::FrameResult, 86
 results_mutex
 QueuedCPUTracker, 171
 rm_create
 lv_resultmanager_api.cpp, 309
 rm_destroy
 lv_resultmanager_api.cpp, 309
 rm_destroy_all
 lv_resultmanager_api.cpp, 309
 rm_flush
 lv_resultmanager_api.cpp, 309
 rm_getbeadresults
 lv_resultmanager_api.cpp, 309
 rm_getconfig
 lv_resultmanager_api.cpp, 310
 rm_getframecounters
 lv_resultmanager_api.cpp, 310
 rm_getresults
 lv_resultmanager_api.cpp, 310
 rm_instances
 lv_resultmanager_api.cpp, 311
 rm_removebead
 lv_resultmanager_api.cpp, 310
 rm_set_tracker
 lv_resultmanager_api.cpp, 310
 rm_store_frame_info
 lv_resultmanager_api.cpp, 311
 roi
 BeadFinder::Config, 49
 round
 cpu_tracker.cpp, 286
 row
 Matrix3X3, 129

rows
 Image4DMemory, 96
RunCOMAndQI
 main.cpp, 251
RunTest
 main.cpp, 252
RunTracker
 SharedTests.h, 269
RunTrackerResults, 206
 computeStats, 206
 meanErr, 207
 output, 207
 stdev, 207
 truepos, 207
RunZTrace
 main.cpp, 252
RunningVistaOrBetter
 Threads, 223

s00
 LsqSqQuadFit::Coeff, 45
s01
 LsqSqQuadFit::Coeff, 45
s10
 LsqSqQuadFit::Coeff, 45
s11
 LsqSqQuadFit::Coeff, 45
s20
 LsqSqQuadFit::Coeff, 45
s21
 LsqSqQuadFit::Coeff, 45
s30
 LsqSqQuadFit::Coeff, 45
s40
 LsqSqQuadFit::Coeff, 45
S_MUL
 kissfft, 109
SFFT_BOTH
 cpu_tracker.cpp, 285
 simplefft.h, 384
SNPRINTF
 std_incl.h, 321
SPrintf
 utils.cpp, 333
 utils.h, 348
STRCASECMP
 std_incl.h, 321
STRNCASECMP
 std_incl.h, 321
SampleFisherMatrix, 207
 Compute, 208
 ComputeAverageFisher, 208
 FisherElem, 209
 ImgDeriv, 209
 maxValue, 209
 SampleFisherMatrix, 207
SaveImage
 CPUTracker, 65
SaveRow
 BinaryResultFile, 42
 ResultFile, 197
 TextResultFile, 217
SaveSection
 ResultManager, 201
scalar_t
 scalar_types.h, 320
scalar_type
 kissfft, 105
 kissfft_utils::traits, 227
scalar_types.h
 complex_t, 320
 scalar_t, 320
scaling
 ResultManagerConfig, 205
ScatterBiasArea
 main.cpp, 255
sched_cpu
 SpeedInfo, 212
sched_gpu
 SpeedInfo, 212
ScheduleFrame
 QueuedTracker, 193
ScheduleImageData
 QueuedTracker, 194
ScheduleLocalization
 QueuedCPUTracker, 165
 QueuedCUDATracker, 183
 QueuedTracker, 194
schedulingThread
 QueuedCUDATracker, 188
SchedulingThreadEntryPoint
 QueuedCUDATracker, 184
SchedulingThreadMain
 QueuedCUDATracker, 184
ScopedCPUProfiler, 210
 ~ScopedCPUProfiler, 210
 ScopedCPUProfiler, 210
 start, 210
 time, 210
SearchArea
 BeadFinder.cpp, 283
SelectTests
 main.cpp, 255
set
 Atomic, 38
 TImageData, 226
set_ZLUT
 lv_cputrack_api.cpp, 299
set_image_float
 lv_cputrack_api.cpp, 298
set_image_from_memory
 lv_cputrack_api.cpp, 298
set_image_u16
 lv_cputrack_api.cpp, 298
set_image_u8
 lv_cputrack_api.cpp, 298
SetBackgroundPriority

Threads, 223
SetCUDADevices
 QueuedCPUTracker.cpp, 315
 QueuedCUDATracker.cu, 382
 QueuedTracker.h, 318
SetConfigValue
 QueuedCPUTracker, 166
 QueuedCUDATracker, 184
 QueuedTracker, 194
SetImage
 CPUTracker, 65
SetImage16Bit
 CPUTracker, 65
SetImage8Bit
 CPUTracker, 65
SetImageFloat
 CPUTracker, 66
SetImageZLUT
 QueuedCPUTracker, 166
 QueuedTracker, 195
SetLVString
 LabVIEW datatypes and helper functions, 21
SetLocalizationMode
 QueuedCPUTracker, 166
 QueuedCUDATracker, 185
 QueuedTracker, 195
SetPixelCalibrationFactors
 QueuedCPUTracker, 166
 QueuedCUDATracker, 185
 QueuedTracker, 195
SetPixelCalibrationImages
 QueuedCPUTracker, 167
 QueuedCUDATracker, 185
 QueuedCUDATracker::Device, 76
 QueuedTracker, 195
SetRadialWeights
 CPUTracker, 66
 QueuedCPUTracker, 167
 QueuedCUDATracker, 186
 QueuedCUDATracker::Device, 76
 QueuedTracker, 195
SetRadialZLUT
 CPUTracker, 66
 QueuedCPUTracker, 168
 QueuedCUDATracker, 186
 QueuedCUDATracker::Device, 76
 QueuedTracker, 195
SetTracker
 ResultManager, 201
SetTrackerImage
 QueuedCPUTracker, 168
SetZLUTBiasCorrection
 QueuedTracker, 195
sfft, 35
 fft, 35
 fft_forward, 35
 fft_inverse, 35
 fill_twiddles, 36
 swap, 36
sfft::complex
 complex, 46
 conjugate, 46
 imag, 46
 operator*, 47
 operator*=, 47
 operator+, 47
 operator+=, 47
 operator-, 47
 real, 47
 x, 48
 y, 48
sfft::complex< T >, 45
SharedTests.h
 EnableGainCorrection, 266
 FetchResults, 266
 Gauss2DTest, 267
 linspace, 268
 logspace, 268
 MeanStDevError, 268
 RandomFill, 268
 ResampleLUT, 269
 ResizeLUT, 269
 RunTracker, 269
 SpeedAccTest, 270
 TestCMOSNoiseInfluence, 271
 WaitForFinish, 273
ShowCUDAError
 test.cu, 364
similarity
 BeadFinder::Config, 49
SimpleKernel
 test.cu, 364
SimpleTest
 main.cpp, 256
simplefft.h
 SFFT_BOTH, 384
size
 device_vec, 80
 pinned_array, 139
Skew
 main.cpp, 242
SkewImage
 testutils.cpp, 276
 testutils.h, 281
SkewParam
 main.cpp, 257
Sleep
 Threads, 223
SmallImageTest
 main.cpp, 257
SmallestPowerOfTwo
 test.cu, 364
speed
 SpeedAccResult, 211
speed_cpu
 SpeedInfo, 212

speed_gpu
 SpeedInfo, 212
SpeedAccResult, 210
 acc, 211
 bias, 211
 Compute, 211
 crlb, 211
 speed, 211
SpeedAccTest
 SharedTests.h, 270
SpeedCompareTest
 test.cu, 365
SpeedInfo, 211
 sched_cpu, 212
 sched_gpu, 212
 speed_cpu, 212
 speed_gpu, 212
SpeedTest
 main.cpp, 257
 test.cu, 365
sq
 testutils.h, 281
 utils.cpp, 333
sqrt
 std_incl.h, 322
srclImage
 CPUTracker, 67
Start
 QueuedCPUTracker, 168
start
 ScopedCPUProfiler, 210
startFrame
 ResultManager::FrameCounters, 85
State
 QueuedCUDATracker::Stream, 213
state
 QueuedCUDATracker::Stream, 215
status
 ErrorCluster, 82
std_incl.h
 __CRT_SECURE_NO_WARNINGS, 321
 ALLOCA_ARRAY, 321
 ALLOCA, 321
 SNPRINTF, 321
 STRCASECMP, 321
 STRNCASECMP, 321
 sqrt, 322
 uchar, 322
 uint, 322
 ulong, 322
 ushort, 322
 VSNPRINTF, 321
 vector2d, 322
 vector2f, 322
 vector3d, 322
 vector3f, 322
StdDeviation
 utils.h, 349
stdev
 CPUTracker, 67
 RunTrackerResults, 207
StoreFrameInfo
 ResultManager, 201
StoreResult
 ResultManager, 201
Stream
 QueuedCUDATracker::Stream, 213
stream
 QI::StreamInstance, 216
 QueuedCUDATracker::Stream, 215
StreamExecuting
 QueuedCUDATracker::Stream, 213
StreamIdle
 QueuedCUDATracker::Stream, 213
StreamPendingExec
 QueuedCUDATracker::Stream, 213
StreamUpdateZLUTSize
 QueuedCUDATracker, 186
streams
 QueuedCUDATracker, 188
sum_diff
 cpu_tracker.cpp, 286
swap
 sfft, 36
TImageData
 alloc, 225
 at, 225
 copyTo, 225
 data, 227
 free, 225
 h, 227
 interpolate, 225
 interpolate1D, 225
 mean, 226
 normalize, 226
 numPixels, 226
 operator[], 226
 pitch, 226
 set, 226
 TImageData, 225
 w, 227
 writeAsCSV, 226
TImageData< T >, 224
TRK_PROFILE
 QueuedCUDATracker.cu, 382
test
 Matrix3X3, 129
test.cu
 BasicQTrkTest, 353
 BasicQTrkTest_RM, 354
 BenchmarkParams, 355
 BuildZLUT, 356
 check_arg, 357
 check_strarg, 357
 CmdLineRun, 357
 cmp_cpu_qi_fft_out, 368

cmp_cpu_qi_prof, 368
 cmp_gpu_qi_fft_out, 368
 cmp_gpu_qi_prof, 368
 CompareAccuracy, 359
 compute, 360
 cudaSharedMem, 368
 emptyKernel, 360
 getPath, 360
 listDevices, 360
 main, 360
 mul_conjugate, 361
 NearestPowerOfTwo, 361
 ProfileSpeedVsROI, 361
 QICompare, 362
 QTrkCompareTest, 362
 ShowCUDAError, 364
 SimpleKernel, 364
 SmallestPowerOfTwo, 364
 SpeedCompareTest, 365
 SpeedTest, 365
 TestAsync, 366
 TestBenchmarkLUT, 367
 TestGauss2D, 367
 TestRadialLUTGradientMethod, 367
 TestSharedMem, 367
 testWithGlobal, 367
 testWithShared, 368
 test_array_passing
 lv_queuetrk_api.cpp, 308
 TestAsync
 test.cu, 366
 TestBSplineMax
 main.cpp, 260
 TestBackground
 main.cpp, 259
 TestBenchmarkLUT
 test.cu, 367
 TestBias
 main.cpp, 259
 TestBoundCheck
 main.cpp, 260
 TestCMOSNoiseInfluence
 SharedTests.h, 271
 TestFourierLUTOnDataset
 main.cpp, 261
 TestFourierLUT
 main.cpp, 260
 TestGauss2D
 test.cu, 367
 TestInterference
 main.cpp, 261
 TestQuadrantAlign
 main.cpp, 261
 TestROIDisplacement
 main.cpp, 262
 TestRadialLUTGradientMethod
 test.cu, 367
 testRun
 CPUTracker, 67
 QTrkSettings, 151
 TestSharedMem
 test.cu, 367
 TestSkew
 main.cpp, 262
 testWithGlobal
 test.cu, 367
 testWithShared
 test.cu, 368
 TestZLUTAlign
 main.cpp, 262
 TestZRange
 main.cpp, 263
 TestZRangeBias
 main.cpp, 265
 Tests
 main.cpp, 242
 testutils.cpp
 AddImages, 274
 BackgroundMedian, 274
 BackgroundRMS, 274
 BackgroundStdDev, 274
 CropImage, 274
 DirExists, 275
 distance, 275
 GaussMask, 275
 GetOuterEdges, 275
 NumFilesInDir, 275
 NumJpgInDir, 276
 ResizeImage, 276
 SkewImage, 276
 testutils.h
 AddImages, 278
 BackgroundMedian, 278
 BackgroundRMS, 278
 BackgroundStdDev, 278
 Console, 277
 CropImage, 278
 DirExists, 279
 distance, 279
 Files, 277
 GaussMask, 279
 GetOuterEdges, 279
 Images, 277
 NumFilesInDir, 280
 NumJpgInDir, 280
 OutputModes, 277
 ResizeImage, 280
 SkewImage, 281
 sq, 281
 TextResultFile, 217
 f, 218
 LoadRow, 217
 SaveRow, 217
 TextResultFile, 217
 Thread
 QueuedCPUTracker::Thread, 218

thread
 QueuedCPUTracker::Thread, 219
 ResultManager, 204
ThreadCaller
 Threads, 223
ThreadEntryPoint
 ThreadPool, 221
 Threads, 223
ThreadHandle
 threads.h, 323
threadID
 Threads::Handle, 87
ThreadLoop
 ResultManager, 202
ThreadPool
 ~ThreadPool, 220
 AddWork, 220
 GetNewItem, 220
 inProgress, 222
 IsDone, 220
 ItemDone, 220
 ProcessArray, 221
 Quit, 221
 quit, 222
 ThreadEntryPoint, 221
 ThreadPool, 220
 threads, 222
 WaitUntilDone, 221
 work, 222
 workMutex, 222
 worker, 222
ThreadPool< TWorkItem, TFunctor >, 219
Threads, 222
 Create, 223
 GetCPUCount, 223
 RunningVistaOrBetter, 223
 SetBackgroundPriority, 223
 Sleep, 223
 ThreadCaller, 223
 ThreadEntryPoint, 223
 WaitAndClose, 224
threads
 QI, 143
 QueuedCPUTracker, 171
 QueuedCUDATracker, 186
 ThreadPool, 222
threads.h
 parallel_for, 323
 ThreadHandle, 323
Threads::Handle, 87
 callback, 87
 param, 87
 threadID, 87
 winhdl, 87
Threads::Mutex, 130
 ~Mutex, 131
 h, 132
 lock, 131
 lockCount, 132
 msg, 131
 Mutex, 131
 name, 132
 trace, 132
 unlock, 131
time
 MeasureTime, 130
 QueuedCUDATracker, 188
 ScopedCPUProfiler, 210
timestamp
 LocalizationJob, 111
 ResultManager::FrameResult, 86
toVector
 device_vec, 80
totalImg
 Image4DMemory, 96
totalNumBytes
 cudaImageList, 74
totalNumPixels
 cudaImageList, 74
trace
 Threads::Mutex, 132
tracker
 QueuedCPUTracker::Thread, 219
trackerID
 CPUTracker, 67
trackerList
 lv_queuetrk_api.cpp, 308
trackerListMutex
 lv_queuetrk_api.cpp, 308
trackerMutex
 ResultManager, 204
traits_type
 kissfft, 105
transform
 kissfft, 109
tritable
 ZLUTParams, 235
tritableSize
 QIParams, 144
truepos
 RunTrackerResults, 207
uchar
 std_incl.h, 322
uint
 std_incl.h, 322
ulong
 std_incl.h, 322
unbind
 cudaImageList, 74
 Image4DCudaArray, 92
 Image4DMemory, 95
UnbindTexture
 ImageSampler_InterpolatedTexture, 98
 ImageSampler_MemCopy, 99
 ImageSampler_SimpleTextureRead, 100
unlock

QueuedCPUTracker::Thread, 218
 Threads::Mutex, 131
Update
 QTrkComputedConfig, 146
 ResultManager, 202
UpdateZLUTs
 QueuedCPUTracker, 168
useTextureCache
 QueuedCUDATracker, 188
ushort
 std_incl.h, 322
utils.cpp
 ApplyGaussianNoise, 324
 ApplyPoissonNoise, 324
 ComputeBgCorrectedCOM1D, 325
 ComputeCRP, 325
 ComputeRadialBinWindow, 326
 ComputeRadialProfile, 326
 CopyImageToFloat, 327
 dbgout, 327
 dbgprintf, 327
 dbgsetlogfile, 327
 file_ext, 328
 GenerateGaussianSpotImage, 328
 GenerateImageFromLUT, 328
 GenerateTestImage, 329
 GetCurrentOutputPath, 329
 GetDirectoryFromPath, 329
 GetFormattedTimeString, 330
 GetLocalModuleFilename, 330
 GetLocalModulePath, 330
 GetPreciseTime, 330
 logFilename, 335
 NearestPowerOf2, 330
 NearestPowerOf3, 331
 NormalizeRadialProfile, 331
 NormalizeZLUT, 331
 ReadCSV, 331
 ReadJPEGFile, 332
 ReadLUTFile, 332
 ReadToByteBuffer, 332
 ReadVector3CSV, 333
 SPrintf, 333
 sq, 333
 WriteArrayAsCSVRow, 333
 WriteComplexImageAsCSV, 333
 WriteImageAsCSV, 334
 WriteToLog, 334
 WriteTrace, 334
 WriteVectorAsCSVRow, 335
utils.h
 ApplyGaussianNoise, 337
 ApplyPoissonNoise, 337
 ComputeBgCorrectedCOM1D, 337
 ComputeCRP, 338
 ComputeRadialBinWindow, 338
 ComputeRadialProfile, 339
 ComputeStdDev, 339
 dbgout, 340
 dbgprintf, 340
 dbgsetlogfile, 340
 DeleteAllElems, 340
 erf, 340
 file_ext, 340
 FloatToJPEGFile, 341
 floatToNormalizedInt, 341
 GenerateGaussianSpotImage, 341
 GenerateImageFromLUT, 341
 GenerateTestImage, 342
 GetCurrentOutputPath, 343
 GetDirectoryFromPath, 343
 GetFormattedTimeString, 343
 GetLocalModuleFilename, 343
 GetLocalModulePath, 343
 GetPreciseTime, 344
 ImageData, 337
 ImageDataad, 337
 Interpolate, 344
 Interpolate1D, 344
 isNaN, 344
 Lerp, 344
 NearestPowerOf2, 345
 NearestPowerOf3, 345
 normalize, 345
 NormalizeRadialProfile, 345
 NormalizeZLUT, 345
 qselect, 346
 ReadCSV, 346
 ReadJPEGFile, 346, 347
 ReadLUTFile, 348
 ReadToByteBuffer, 348
 ReadVector3CSV, 348
 SPrintf, 348
 StdDeviation, 349
 WriteArrayAsCSVRow, 349
 WriteComplexImageAsCSV, 349
 WriteImageAsCSV, 349
 WriteJPEGFile, 350, 351
 WriteTrace, 351
 WriteVectorAsCSVRow, 351
VSNPRINTF
 std_incl.h, 321
ValidRM
 lv_resultmanager_api.cpp, 311
ValidateTracker
 LabVIEW datatypes and helper functions, 22
vector2
 random, 229
 vector2, 228
 x, 229
 y, 229
 vector2< T >, 228
vector2d
 std_incl.h, 322
vector2f
 std_incl.h, 322

vector3
length, 230
operator*, 230, 232
operator*=
operator+, 231
operator+=
operator-, 231
operator-=, 232
operator/, 232
operator/=, 232
vector3, 230
x, 233
xy, 232
y, 233
z, 233
vector3< T >, 229
vector3d
 std_incl.h, 322
vector3f
 std_incl.h, 322
vertexForm
 LsqSqQuadFit, 116

w
 cudaImageList, 74
 ImageLUTConfig, 97
 TImageData, 227

WRITEVAR
 QueuedTracker.cpp, 316

WaitAndClose
 Threads, 224

WaitForFinish
 SharedTests.h, 273

WaitUntilDone
 ThreadPool, 221

width
 CPUTracker, 67
 FFT2DTracker, 84
 QTrkSettings, 152

winhdl
 Threads::Handle, 87

work
 ThreadPool, 222

workMutex
 ThreadPool, 222

worker
 ThreadPool, 222

WorkerThreadMain
 QueuedCPUTracker, 169

Write
 ResultManager, 202

write
 Image4DMemory, 96

WriteArrayAsCSVRow
 utils.cpp, 333
 utils.h, 349

writeAsCSV
 TImageData, 226

WriteBinaryFileHeader

 ResultManager, 203

WriteBinaryResults
 ResultManager, 203

WriteComplexImageAsCSV
 utils.cpp, 333
 utils.h, 349

WriteImageAsCSV
 utils.cpp, 334
 utils.h, 349

writeInterval
 ResultManagerConfig, 205

WriteJPEGFile
 fastjpg.cpp, 290
 utils.h, 350, 351

WriteRadialProf
 main.cpp, 265

writeSurfacePixel
 Image4DCudaArray::KernelInst, 102

WriteTextResults
 ResultManager, 204

WriteToFile
 QTrkComputedConfig, 146

WriteToLog
 QTrkComputedConfig, 146
 utils.cpp, 334

WriteTrace
 utils.cpp, 334
 utils.h, 351

WriteVectorAsCSVRow
 utils.cpp, 335
 utils.h, 351

x
 BeadFinder::Position, 140
 ROIPosition, 206
 sfft::complex, 48
 vector2, 229
 vector3, 233

XCor1DBuffer, 233
 fft_backward, 234
 fft_forward, 234
 XCor1DBuffer, 233
 XCorFFTHelper, 233
 xcorw, 234

XCorFFTHelper
 XCor1DBuffer, 233

XCorScale
 cpu_tracker.cpp, 286

xc1_iterations
 QTrkSettings, 152

xc1_profileLength
 QTrkSettings, 152

xc1_profileWidth
 QTrkSettings, 152

xcorBuffer
 CPUTracker, 67

xcorw
 CPUTracker, 67
 XCor1DBuffer, 234

xfft
 CPUTracker::FFT2D, 83
 xoffset
 LsqSqQuadFit, 117
 xscale
 ImageLUTConfig, 97
 xy
 LVArray2D, 118
 vector3, 232

 y
 BeadFinder::Position, 140
 ROIPosition, 206
 sfft::complex, 48
 vector2, 229
 vector3, 233
 yfft
 CPUTracker::FFT2D, 83
 yscale
 ImageLUTConfig, 97

 z
 vector3, 233
 ZLUT_CMPDATA
 cpu_tracker.cpp, 285
 ZLUT_ComputeProfileMatchScores
 Kernels.h, 374
 ZLUT_ComputeZ
 Kernels.h, 375
 ZLUT_LSQFIT_NWEIGHTS
 QueuedTracker.h, 317
 ZLUT_LSQFIT_WEIGHTS
 QueuedTracker.h, 317
 ZLUT_NormalizeProfiles
 Kernels.h, 375
 ZLUT_ProfilesToZLUT
 Kernels.h, 375
 ZLUT_RadialProfileKernel
 Kernels.h, 376
 ZLUTBiasCorrection
 QueuedTracker, 195
 ZLUTParams, 234
 angularSteps, 235
 GetRadialZLUT, 234
 img, 235
 maxRadius, 235
 minRadius, 235
 planes, 235
 radialSteps, 234
 tritable, 235
 zcmpwindow, 235
 ZLUTSelfTest
 QueuedCPUTracker, 169
 ZLUTWeights
 cpu_tracker.cpp, 286
 ZLUTWeights_d
 cpu_tracker.cpp, 286
 zcmp
 QueuedCPUTracker, 171

 zcmpwindow
 ZLUTParams, 235
 zcompareWindow
 QueuedCUDATracker::Device, 77
 zcompute
 QueuedCUDATracker::KernelProfileTime, 104
 zcomputeDone
 QueuedCUDATracker::Stream, 215
 zlut_angular_coverage
 QTrkSettings, 152
 zlut_angularsteps
 QTrkComputedConfig, 147
 zlut_bias_correction
 QueuedTracker, 196
 zlut_build_flags
 QueuedCUDATracker, 188
 zlut_buildflags
 QueuedCPUTracker, 171
 zlut_cmpprofiles
 QueuedCPUTracker, 171
 zlut_count
 CPUTracker, 67
 QueuedCPUTracker, 171
 zlut_enablecmpprof
 QueuedCPUTracker, 171
 zlut_maxradius
 CPUTracker, 67
 QTrkComputedConfig, 147
 zlut_memoryOwner
 CPUTracker, 67
 zlut_minradius
 CPUTracker, 67
 QTrkSettings, 152
 zlut_planes
 CPUTracker, 67
 QueuedCPUTracker, 171
 zlut_radial_coverage
 QTrkSettings, 152
 zlut_radialsteps
 QTrkComputedConfig, 147
 zlut_radialweights
 CPUTracker, 67
 zlut_res
 CPUTracker, 67
 zlut_roi_coverage
 QTrkSettings, 152
 zlut_tritable
 QueuedCUDATracker::Device, 77
 zlut_useCorrelation
 CPUTracker, 67
 zlutAlign
 QueuedCUDATracker::KernelProfileTime, 104
 zlutIndex
 LocalizationJob, 111
 LocalizationParams, 112
 zlutPlane
 LocalizationParams, 112
 zluts

CPUTracker, [67](#)
QueuedCPUTracker, [171](#)