Costa’s tools

Generated by Doxygen 1.8.1.2

Wed Dec 12 2012 14:37:38

Contents

1 Namespace Index **1**

1.1 Namespace List 1

2 Class Index **3**

2.1 Class List 3

3 File Index **5**

3.1 File List 5

4 Namespace Documentation **7**

4.1 CommonTools Namespace Reference 7

4.1.1 Enumeration Type Documentation 9

4.1.1.1 VTKSurfaceMeshFormats 9

4.1.1.2 VTKVolumeMeshFormats 9

4.1.2 Function Documentation 9

4.1.2.1 CheckSaveFileExtension 9

4.1.2.2 CloseSurface 9

4.1.2.3 ExportPolyDataPoints 9

4.1.2.4 ExportPolyDataPoints 9

4.1.2.5 FileExists 9

4.1.2.6 GenerateDecimationScript 10

4.1.2.7 GenerateHoleCover 10

4.1.2.8 GetP2P 10

4.1.2.9 GetP2S 10

4.1.2.10 GetP2S 10

4.1.2.11 GetP2S 10

4.1.2.12 GetS2S 10

4.1.2.13 GetShapeSubSurface 10

4.1.2.14 GetShapeSubSurface 10

4.1.2.15 GetTypeOfVTKData 10

4.1.2.16 GetTypeOfVTKVolumeData 11

4.1.2.17 ImportPolyDataPoints 11

4.1.2.18 ImportPolyDataPoints 11
4.1.2.19 laplace3D_voxelsize 11
4.1.2.20 LoadImage 11
4.1.2.21 LoadShapeFromFile 11
4.1.2.22 LoadVolumeFromFile 11
4.1.2.23 LoadVtkShortArray 11
4.1.2.24 Points2Polydata 11
4.1.2.25 Points2Polydata 12
4.1.2.26 ReadFilelist 12
4.1.2.27 SaveImage 12
4.1.2.28 SavePoints 12
4.1.2.29 SavePolydata 12
4.1.2.30 SaveShapeToFile 12
4.1.2.31 SaveUnstructuredGrid 12
4.1.2.32 SaveVolumeToFile 12
4.1.2.33 SaveVtkShortArray 12
4.1.2.34 ScaleShape 13
4.1.2.35 ScaleVolume 13
4.1.2.36 ShrinkImage 13
4.2 itk Namespace Reference 13

5 Class Documentation **15**
5.1 itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage > Class Template Reference . . . 15
5.1.1 Detailed Description 16
5.1.2 Member Typedef Documentation 16
5.1.2.1 ConstBoundaryConditionType 16
5.1.2.2 ConstPointer 16
5.1.2.3 IndexType 16
5.1.2.4 InputImagePixelType 17
5.1.2.5 InputImagePointer 17
5.1.2.6 InputImageType 17
5.1.2.7 NeighborhoodIteratorType 17
5.1.2.8 NeighborhoodType 17
5.1.2.9 OutputImagePixelType 17
5.1.2.10 OutputImagePointer 17
5.1.2.11 OutputImageType 17
5.1.2.12 Pointer 17
5.1.2.13 RegionType 17
5.1.2.14 Self 18
5.1.2.15 SizeType 18

5.1.2.16 Superclass 18
5.1.3 Constructor & Destructor Documentation 18
5.1.3.1 BinaryThinningImageFilter3D 18
5.1.3.2 ∼BinaryThinningImageFilter3D 18
5.1.4 Member Function Documentation 18
5.1.4.1 ComputeThinImage 18
5.1.4.2 fillEulerLUT 18
5.1.4.3 GenerateData 18
5.1.4.4 GetThinning 18
5.1.4.5 isEulerInvariant 18
5.1.4.6 isSimplePoint 18
5.1.4.7 itkNewMacro 19
5.1.4.8 itkStaticConstMacro 19
5.1.4.9 itkStaticConstMacro 19
5.1.4.10 itkTypeMacro 19
5.1.4.11 Octree_labeling 19
5.1.4.12 PrepareData 19
5.1.4.13 PrintSelf 19
5.2 BITMAPFILEHEADER Struct Reference 19
5.2.1 Member Data Documentation 19
5.2.1.1 DataOffset 19
5.2.1.2 FileSize 19
5.2.1.3 reserved 20
5.2.1.4 Signature 20
5.3 BITMAPINFOHEADER Struct Reference 20
5.3.1 Member Data Documentation 20
5.3.1.1 BitCount 20
5.3.1.2 ColorsImportant 20
5.3.1.3 ColorsUsed 20
5.3.1.4 Compression 20
5.3.1.5 Height 20
5.3.1.6 ImageSize 20
5.3.1.7 Planes 20
5.3.1.8 Size 20
5.3.1.9 Width 20
5.3.1.10 XpixelsPerM 20
5.3.1.11 YpixelsPerM 20
5.4 dcomplex Struct Reference 21
5.4.1 Detailed Description 21
5.4.2 Member Data Documentation 21

5.4.2.1 im 21

5.4.2.2 re 21

5.5 fcomplex Struct Reference 21

5.5.1 Detailed Description 21

5.5.2 Member Data Documentation 22

5.5.2.1 im 22

5.5.2.2 re 22

5.6 itk::ImageToVTKImageFilter< TInputImage > Class Template Reference 22

5.6.1 Detailed Description 23

5.6.2 Member Typedef Documentation 23

5.6.2.1 ConstPointer 23

5.6.2.2 ExporterFilterPointer 23

5.6.2.3 ExporterFilterType 23

5.6.2.4 InputImagePointer 23

5.6.2.5 InputImageType 23

5.6.2.6 Pointer 23

5.6.2.7 Self 23

5.6.2.8 Superclass 23

5.6.3 Constructor & Destructor Documentation 23

5.6.3.1 ImageToVTKImageFilter 23

5.6.3.2 ∼ImageToVTKImageFilter 23

5.6.4 Member Function Documentation 23

5.6.4.1 addtest 23

5.6.4.2 addvector 24

5.6.4.3 GetExporter 24

5.6.4.4 GetImporter 24

5.6.4.5 GetOutput 24

5.6.4.6 getvtest 24

5.6.4.7 itkNewMacro 24

5.6.4.8 itkTypeMacro 24

5.6.4.9 SetInput 24

5.6.4.10 testsize 24

5.6.4.11 tralala 24

5.6.4.12 Update 24

5.7 RGBFILEHEADER Struct Reference 25

5.7.1 Member Data Documentation 25

5.7.1.1 bytespercha 25

5.7.1.2 cmaptype 25

5.7.1.3 components 25

5.7.1.4 compression 25

5.7.1.5 dim . . . 25
5.7.1.6 dummy . . . 25
5.7.1.7 height . . . 25
5.7.1.8 magic . . . 25
5.7.1.9 maxcol . . . 25
5.7.1.10 mincol . . . 25
5.7.1.11 name . . . 25
5.7.1.12 width . . . 25
5.8 sorting_struct Struct Reference . . . 25
5.8.1 Member Data Documentation . . . 26
5.8.1.1 index . . . 26
5.8.1.2 value . . . 26
5.9 xvimage Struct Reference . . . 26
5.9.1 Detailed Description . . . 27
5.9.2 Member Data Documentation . . . 27
5.9.2.1 col_size . . . 27
5.9.2.2 d . . . 27
5.9.2.3 data_storage_type . . . 27
5.9.2.4 depth_size . . . 27
5.9.2.5 image_data . . . 27
5.9.2.6 name . . . 27
5.9.2.7 num_data_bands . . . 27
5.9.2.8 origin_x . . . 27
5.9.2.9 origin_y . . . 27
5.9.2.10 origin_z . . . 27
5.9.2.11 row_size . . . 27
5.9.2.12 time_size . . . 28
5.9.2.13 xdim . . . 28
5.9.2.14 xmax . . . 28
5.9.2.15 xmin . . . 28
5.9.2.16 ydim . . . 28
5.9.2.17 ymax . . . 28
5.9.2.18 ymin . . . 28
5.9.2.19 zdim . . . 28
5.9.2.20 zmax . . . 28
5.9.2.21 zmin . . . 28

6 File Documentation 29

6.1 /home/costa/Data/Code/C/apps/Blob.cpp File Reference . . . 29
6.1.1 Detailed Description . . . 29

6.1.2 Function Documentation 29
6.1.2.1 ApproximateCurvature 30
6.1.2.2 GetPointNeighbors 30
6.1.2.3 main 30
6.2 /home/costa/Data/Code/C/apps/chaste2vtk.cpp File Reference 30
6.2.1 Detailed Description 30
6.2.2 Function Documentation 30
6.2.2.1 main 30
6.2.2.2 ReadChasteElements 30
6.2.2.3 ReadChasteNodes 30
6.3 /home/costa/Data/Code/C/apps/CloseBVMesh.cpp File Reference 31
6.3.1 Detailed Description 31
6.3.2 Function Documentation 31
6.3.2.1 CopyCellScalars 31
6.3.2.2 FillSmallHoles 31
6.3.2.3 main 32
6.3.2.4 PointInPolygon 32
6.3.2.5 PolyData2Polygon 32
6.3.2.6 PolygonBoundaryArea 32
6.4 /home/costa/Data/Code/C/apps/CommonTools.cpp File Reference 33
6.5 /home/costa/Data/Code/C/apps/CommonTools.h File Reference 34
6.5.1 Detailed Description 36
6.6 /home/costa/Data/Code/C/apps/CreateImageMask.cpp File Reference 36
6.6.1 Detailed Description 36
6.6.2 Function Documentation 36
6.6.2.1 main 36
6.7 /home/costa/Data/Code/C/apps/ExtractShapeRegion.cpp File Reference 36
6.7.1 Detailed Description 37
6.7.2 Function Documentation 37
6.7.2.1 main 37
6.8 /home/costa/Data/Code/C/apps/GenerateVolumetricLVMesh.cpp File Reference 37
6.8.1 Detailed Description 39
6.8.2 Macro Definition Documentation 39
6.8.2.1 FIELD_DT 39
6.8.2.2 FIELD_LAPLACE 39
6.8.3 Function Documentation 39
6.8.3.1 GenerateImageMask 39
6.8.3.2 GenerateLayersAlongField 39
6.8.3.3 GenerateLayersAlongNormals 39
6.8.3.4 GenerateLocalCoordinateCircLongit 39

6.8.3.5 GenerateLocalCoordinateRadial 39
6.8.3.6 main 39
6.8.3.7 usage 39
6.9 /home/costa/Data/Code/C/apps/imported/itkBinaryThinningImageFilter3D.h File Reference 39
6.10 /home/costa/Data/Code/C/apps/imported/itkImageToVTKImageFilter.h File Reference 40
6.11 /home/costa/Data/Code/C/apps/imported/LabelBranches3D.cpp File Reference 40
6.11.1 Macro Definition Documentation 41
6.11.1.1 FLT_MAX 41
6.11.2 Function Documentation 41
6.11.2.1 check_neighborhood 41
6.11.2.2 check_num_connected_neighbors 41
6.11.2.3 CheckConnect 41
6.11.2.4 CheckPointState 41
6.11.2.5 comp_conexas 41
6.11.2.6 distance 41
6.11.2.7 find_maximun 41
6.11.2.8 find_shorter_branch 42
6.11.2.9 findmaximum_centerline 42
6.11.2.10 FindTriplePoints 42
6.11.2.11 LabelBranchs 42
6.11.2.12 LabelBranchsNew 42
6.11.2.13 main 42
6.11.2.14 mapIndex3D 42
6.11.2.15 maptox 42
6.11.2.16 maptoy 42
6.11.2.17 maptoz 42
6.11.2.18 print_timing 42
6.11.2.19 Relabeling 42
6.11.2.20 reserve_memory_triple_float 42
6.11.2.21 reserve_memory_triple_int 42
6.11.3 Variable Documentation 42
6.11.3.1 debug 42
6.11.3.2 verbose 42
6.12 /home/costa/Data/Code/C/apps/imported/pgm2itkvol/itkvol2pgm.cxx File Reference 42
6.12.1 Macro Definition Documentation 43
6.12.1.1 VERBOSE 43
6.12.2 Function Documentation 43
6.12.2.1 main 43
6.13 /home/costa/Data/Code/C/apps/imported/pgm2itkvol/mccodimage.h File Reference 43
6.13.1 Detailed Description 45

6.13.2 Macro Definition Documentation 45
6.13.2.1 ACCEPTED_TYPES1 45
6.13.2.2 ACCEPTED_TYPES2 45
6.13.2.3 ACCEPTED_TYPES3 45
6.13.2.4 ACCEPTED_TYPES4 46
6.13.2.5 ACCEPTED_TYPES5 46
6.13.2.6 ACCEPTED_TYPES6 46
6.13.2.7 ACCEPTED_TYPES7 46
6.13.2.8 colsize 46
6.13.2.9 COMPARE_SIZE 46
6.13.2.10 COMPLEXDATA 47
6.13.2.11 datatype 47
6.13.2.12 DCOMPLEXDATA 47
6.13.2.13 depth 47
6.13.2.14 DERRIERE 47
6.13.2.15 DEVANT 47
6.13.2.16 DOUBLEDATA 47
6.13.2.17 EST 47
6.13.2.18 FLOATDATA 47
6.13.2.19 lpixel 47
6.13.2.20 lvoxel 47
6.13.2.21 nbands 47
6.13.2.22 NDG_MAX 47
6.13.2.23 NDG_MIN 47
6.13.2.24 nonbord 47
6.13.2.25 nonbord3d 47
6.13.2.26 NORD 47
6.13.2.27 NORD_EST 47
6.13.2.28 NORD_OUEST 47
6.13.2.29 ONLY_2D 47
6.13.2.30 ONLY_3D 47
6.13.2.31 OUEST 48
6.13.2.32 pixel 48
6.13.2.33 rowsize 48
6.13.2.34 SCHARDATA 48
6.13.2.35 SLONGDATA 48
6.13.2.36 SSHORTDATA 48
6.13.2.37 SUD 48
6.13.2.38 SUD_EST 48
6.13.2.39 SUD_OUEST 48

6.13.2.40 tsize 48
6.13.2.41 UCHARDATA 48
6.13.2.42 ULONGDATA 48
6.13.2.43 USHORTDATA 48
6.13.2.44 VFF_TYP_1_BYTE 48
6.13.2.45 VFF_TYP_2_BYTE 48
6.13.2.46 VFF_TYP_4_BYTE 48
6.13.2.47 VFF_TYP_BIT 48
6.13.2.48 VFF_TYP_COMPLEX 48
6.13.2.49 VFF_TYP_DCOMPLEX 48
6.13.2.50 VFF_TYP_DOUBLE 48
6.13.2.51 VFF_TYP_FLOAT 48
6.13.2.52 voxel 48
6.13.3 Typedef Documentation 48
6.13.3.1 xvimage 48
6.13.4 Function Documentation 48
6.13.4.1 bord 48
6.13.4.2 bord3d 48
6.13.4.3 maskvois26 48
6.13.4.4 sont18voisins 49
6.13.4.5 sont26voisins 49
6.13.4.6 sont4voisins 49
6.13.4.7 sont6voisins 49
6.13.4.8 sont8voisins 49
6.13.4.9 voisin 49
6.13.4.10 voisin125 49
6.13.4.11 voisin14b 49
6.13.4.12 voisin18 49
6.13.4.13 voisin2 49
6.13.4.14 voisin26 49
6.13.4.15 voisin5 49
6.13.4.16 voisin6 49
6.13.4.17 voisin6b 49
6.13.4.18 voisinENAR 49
6.13.4.19 voisinENAV 49
6.13.4.20 voisinNESO 49
6.13.4.21 voisinNOSE 49
6.13.4.22 voisinONAR 49
6.13.4.23 voisinONAV 49
6.13.4.24 voisins18 49

6.13.4.25 voisins26 49
6.13.4.26 voisins4 49
6.13.4.27 voisins6 49
6.13.4.28 voisins8 49
6.14 /home/costa/Data/Code/C/apps/imported/pgm2itkvol/mcimage.c File Reference 50
6.14.1 Macro Definition Documentation 52
6.14.1.1 BUFFERSIZE 52
6.14.1.2 F_NAME 52
6.14.1.3 F_NAME 52
6.14.1.4 F_NAME 52
6.14.1.5 F_NAME 52
6.14.1.6 F_NAME 52
6.14.1.7 F_NAME 52
6.14.1.8 F_NAME 52
6.14.1.9 F_NAME 52
6.14.1.10 F_NAME 52
6.14.1.11 F_NAME 52
6.14.1.12 F_NAME 52
6.14.1.13 F_NAME 52
6.14.1.14 F_NAME 52
6.14.1.15 F_NAME 52
6.14.1.16 F_NAME 52
6.14.1.17 F_NAME 52
6.14.1.18 F_NAME 52
6.14.1.19 F_NAME 52
6.14.1.20 F_NAME 52
6.14.1.21 F_NAME 52
6.14.1.22 F_NAME 52
6.14.1.23 F_NAME 52
6.14.1.24 F_NAME 52
6.14.1.25 F_NAME 52
6.14.1.26 F_NAME 52
6.14.1.27 F_NAME 52
6.14.1.28 F_NAME 53
6.14.1.29 F_NAME 53
6.14.1.30 F_NAME 53
6.14.1.31 F_NAME 53
6.14.1.32 F_NAME 53
6.14.1.33 INT32_MAX 53
6.14.1.34 WARN_HUGE 53

6.14.2 Function Documentation 53
6.14.2.1 allocheader 53
6.14.2.2 allocimage 53
6.14.2.3 allocmultimage 53
6.14.2.4 convertfloat 53
6.14.2.5 convertgen 53
6.14.2.6 convertlong 53
6.14.2.7 copy2image 53
6.14.2.8 copyimage 53
6.14.2.9 equalimages 53
6.14.2.10 freadulong 53
6.14.2.11 freadushort 53
6.14.2.12 freeimage 53
6.14.2.13 fwriteulong 54
6.14.2.14 fwriteushort 54
6.14.2.15 image2list 54
6.14.2.16 list2image 54
6.14.2.17 pink_fopen_read 54
6.14.2.18 pink_fopen_write 54
6.14.2.19 printimage 54
6.14.2.20 razimage 54
6.14.2.21 readbmp 54
6.14.2.22 readheader 54
6.14.2.23 readimage 54
6.14.2.24 readlongimage 54
6.14.2.25 readrgb 54
6.14.2.26 readrgbimage 55
6.14.2.27 readse 55
6.14.2.28 showheader 55
6.14.2.29 writeascimage 55
6.14.2.30 writebmp 55
6.14.2.31 writeimage 55
6.14.2.32 writelist2 55
6.14.2.33 writelist3 55
6.14.2.34 writelongimage 55
6.14.2.35 writerawimage 55
6.14.2.36 writergbascimage 55
6.14.2.37 writergbimage 55
6.14.2.38 writese 55
6.15 /home/costa/Data/Code/C/apps/imported/pgm2itkvol/mcimage.h File Reference 55

6.15.1 Detailed Description 56
6.15.2 Macro Definition Documentation 57
6.15.2.1 __pink__inline 57
6.15.2.2 HUGE_IMAGE_SIZE 57
6.15.3 Typedef Documentation 57
6.15.3.1 index_t 57
6.15.3.2 int32_t 57
6.15.3.3 u_int32_t 57
6.15.3.4 u_int8_t 57
6.15.3.5 uint32_t 57
6.15.3.6 uint8_t 57
6.15.4 Function Documentation 57
6.15.4.1 allocheader 57
6.15.4.2 allocimage 57
6.15.4.3 allocmultimage 57
6.15.4.4 convertfloat 57
6.15.4.5 convertgen 57
6.15.4.6 convertlong 57
6.15.4.7 copy2image 57
6.15.4.8 copyimage 57
6.15.4.9 equalimages 57
6.15.4.10 freeimage 58
6.15.4.11 image2list 58
6.15.4.12 list2image 58
6.15.4.13 printimage 58
6.15.4.14 razimage 58
6.15.4.15 readbmp 58
6.15.4.16 readheader 58
6.15.4.17 readimage 58
6.15.4.18 readlongimage 58
6.15.4.19 readrgb 58
6.15.4.20 readrgbimage 58
6.15.4.21 readse 59
6.15.4.22 showheader 59
6.15.4.23 writeascimage 59
6.15.4.24 writebmp 59
6.15.4.25 writeimage 59
6.15.4.26 writelist2 59
6.15.4.27 writelist3 59
6.15.4.28 writelongimage 59

6.15.4.29 writerawimage 59

6.15.4.30 writergbascimage 59

6.15.4.31 writergbimage 59

6.15.4.32 writese 59

6.16 /home/costa/Data/Code/C/apps/imported/pgm2itkvol/mcutil.h File Reference 59

6.16.1 Macro Definition Documentation 60

6.16.1.1 arrondi 60

6.16.1.2 M_1_PI 60

6.16.1.3 M_2_PI 60

6.16.1.4 M_2_SQRTPI 60

6.16.1.5 M_E 60

6.16.1.6 M_LN10 60

6.16.1.7 M_LN2 60

6.16.1.8 M_LOG10E 60

6.16.1.9 M_LOG2E 60

6.16.1.10 M_PI 60

6.16.1.11 M_PI_2 60

6.16.1.12 M_PI_4 60

6.16.1.13 M_SQRT1_2 60

6.16.1.14 M_SQRT2 60

6.16.1.15 mcabs 60

6.16.1.16 mceven 60

6.16.1.17 mcmax 60

6.16.1.18 mcmin 60

6.16.1.19 mcodd 60

6.16.1.20 mcsqr 60

6.16.1.21 signe 60

6.17 /home/costa/Data/Code/C/apps/imported/pgm2itkvol/pgm2itkvol.cxx File Reference 61

6.17.1 Macro Definition Documentation 61

6.17.1.1 UINTDATA 61

6.17.1.2 VERBOSE 61

6.17.2 Function Documentation 61

6.17.2.1 main 61

6.18 /home/costa/Data/Code/C/apps/LabelBiventricularMesh.cpp File Reference 61

6.18.1 Detailed Description 62

6.18.2 Function Documentation 62

6.18.2.1 main 62

6.18.2.2 SetScalars 62

6.19 /home/costa/Data/Code/C/apps/MakeBiventricularMesh.cpp File Reference 62

6.19.1 Detailed Description 63

6.19.2 Function Documentation 63
6.19.2.1 CreateEmptyImage 63
6.19.2.2 CreateMask 63
6.19.2.3 DecimateMesh 63
6.19.2.4 main 63
6.20 /home/costa/Data/Code/C/apps/MeshHeart.cpp File Reference 63
6.20.1 Detailed Description 64
6.20.2 Function Documentation 64
6.20.2.1 ExtractSurface 64
6.20.2.2 main 64
6.21 /home/costa/Data/Code/C/apps/MeshSegmentationLaplace.cpp File Reference 64
6.21.1 Detailed Description 64
6.21.2 Function Documentation 64
6.21.2.1 main 64
6.22 /home/costa/Data/Code/C/apps/MRIRemesh.cpp File Reference 64
6.22.1 Detailed Description 65
6.22.2 Macro Definition Documentation 66
6.22.2.1 CONTOUR_ENDO 66
6.22.2.2 CONTOUR_ENDO_APICAL 66
6.22.2.3 CONTOUR_ENDO_BASAL 66
6.22.2.4 CONTOUR_EPI 66
6.22.2.5 CONTOUR_EPI_APICAL 66
6.22.2.6 CONTOUR_EPI_BASAL 66
6.22.2.7 REF_POINT 66
6.22.3 Function Documentation 66
6.22.3.1 ArbitraryRotate 66
6.22.3.2 ComputeCentroid 66
6.22.3.3 main 66
6.22.3.4 OrderPoints 66
6.22.3.5 struct_cmp_by_value 66
6.23 /home/costa/Data/Code/C/apps/PassScalars.cpp File Reference 66
6.23.1 Detailed Description 67
6.23.2 Function Documentation 67
6.23.2.1 main 67
6.24 /home/costa/Data/Code/C/apps/PassScalarsInterp.cpp File Reference 67
6.24.1 Detailed Description 67
6.24.2 Function Documentation 67
6.24.2.1 main 67
6.24.2.2 PassScalarsFloat 67
6.25 /home/costa/Data/Code/C/apps/PassScalarsReverse.cpp File Reference 68

6.25.1 Detailed Description 68
6.25.2 Function Documentation 68
6.25.2.1 main 68
6.25.2.2 PassScalarsFloat 68
6.25.2.3 PassScalarsShort 68
6.26 /home/costa/Data/Code/C/apps/ResampleImage.cpp File Reference 68
6.26.1 Detailed Description 69
6.26.2 Function Documentation 69
6.26.2.1 main 69
6.27 /home/costa/Data/Code/C/apps/SetScalars.cpp File Reference 69
6.27.1 Detailed Description 70
6.27.2 Function Documentation 70
6.27.2.1 AddArray 70
6.27.2.2 main 70
6.28 /home/costa/Data/Code/C/apps/SmoothMeshTrhoughImage.cpp File Reference 70
6.28.1 Detailed Description 70
6.28.2 Function Documentation 70
6.28.2.1 main 70
6.29 /home/costa/Data/Code/C/apps/TransformPhilipsHeart.cpp File Reference 70
6.29.1 Detailed Description 71
6.29.2 Function Documentation 71
6.29.2.1 main 71
6.30 /home/costa/Data/Code/C/apps/VTKConvert.cpp File Reference 71
6.30.1 Detailed Description 71
6.30.2 Function Documentation 71
6.30.2.1 main 71

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

CommonTools 7

itk 13

Chapter 2

Class Index

2.1 Class List

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

itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >

This filter computes one-pixel-wide skeleton of a 3D input image 15

BITMAPFILEHEADER 19

BITMAPINFOHEADER 20

dcomplex

Complex number, represented by doubles 21

fcomplex

Complex number, represented by floats 21

itk::ImageToVTKImageFilter< TInputImage >

Converts an ITK image into a VTK image and plugs a itk data pipeline to a VTK datapipeline . 22

RGBFILEHEADER 25

sorting_struct 25

xvimage

The image class for the C functions 26

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

/home/costa/Data/Code/C/apps/Blob.cpp
Active surfaces. An active surfaces implementation that fits a 3D blob to a set of points. Under construction . . . 29
/home/costa/Data/Code/C/apps/chaste2vtk.cpp
Convert chaste .ele .node tetrahedral mesh to VTK unstructured grid (only for Oxford Rabbit) . 30
/home/costa/Data/Code/C/apps/CloseBVMesh.cpp
Closes biventricular mesh by connecting endocardial edge to epicardial at the base. Does not close ventricles . . . 31
/home/costa/Data/Code/C/apps/CommonTools.cpp . . . 33
/home/costa/Data/Code/C/apps/CommonTools.h
Some common functions . . . 34
/home/costa/Data/Code/C/apps/CreateImageMask.cpp
Create mask for a given shape with user defined dimensions . . . 36
/home/costa/Data/Code/C/apps/ExtractShapeRegion.cpp
Extract a part of polydata . . . 36
/home/costa/Data/Code/C/apps/GenerateVolumetricLVMesh.cpp
Generate volumetric mesh of Left Ventricle . . . 37
/home/costa/Data/Code/C/apps/LabelBiventricularMesh.cpp
Generate labels for Rafa's biventricular mesh . . . 61
/home/costa/Data/Code/C/apps/MakeBiventricularMesh.cpp
Make biventricular mesh for Rafa's model . . . 62
/home/costa/Data/Code/C/apps/MeshHeart.cpp
Something to do with biventricular model generation . . . 63
/home/costa/Data/Code/C/apps/MeshSegmentationLaplace.cpp
Mesh segmentation using distance to skeleton along the gradiwent of the solution of the laplacian equation . . . 64
/home/costa/Data/Code/C/apps/MRIRemesh.cpp
From shortaxis contours in MRI + reference point creates a smoother mesh using splines longitudinally . . . 64
/home/costa/Data/Code/C/apps/PassScalars.cpp
Copies scalars from one polydata to another . . . 66
/home/costa/Data/Code/C/apps/PassScalarsInterp.cpp
Copies scalars from one mesh to another. When the source is sparse, for every target point it finds a corresponding cell and interpolates the point scalars . . . 67
/home/costa/Data/Code/C/apps/PassScalarsReverse.cpp
Same as PassScalars, except that the search is done for every point/cell of target, not source! 68
/home/costa/Data/Code/C/apps/ResampleImage.cpp
Resample image, smooth, reconstruct shape . . . 68

/home/costa/Data/Code/C/apps/SetScalars.cpp
Adds scalar array ro polydata and sets its value to a constant 69
/home/costa/Data/Code/C/apps/SmoothMeshTrhoughImage.cpp
Smooth mesh using image mask 70
/home/costa/Data/Code/C/apps/TransformPhilipsHeart.cpp
Transforms Philips mesh to suit EM simulations. Don't remember WTF is that 70
/home/costa/Data/Code/C/apps/VTKConvert.cpp
Convert vtk polydata between formats 71
/home/costa/Data/Code/C/apps/imported/itkBinaryThinningImageFilter3D.h 39
/home/costa/Data/Code/C/apps/imported/itkImageToVTKImageFilter.h 40
/home/costa/Data/Code/C/apps/imported/LabelBranches3D.cpp 40
/home/costa/Data/Code/C/apps/imported/pgm2itkvol/itkvol2pgm.cxx 42
/home/costa/Data/Code/C/apps/imported/pgm2itkvol/mccodimage.h
This file hold the basic type declarations used in the C functions of Pink 43
/home/costa/Data/Code/C/apps/imported/pgm2itkvol/mcimage.c 50
/home/costa/Data/Code/C/apps/imported/pgm2itkvol/mcimage.h
This file holds the basic image allocation functions 55
/home/costa/Data/Code/C/apps/imported/pgm2itkvol/mcutil.h 59
/home/costa/Data/Code/C/apps/imported/pgm2itkvol/pgm2itkvol.cxx 61

Chapter 4

Namespace Documentation

4.1 CommonTools Namespace Reference

Enumerations

- enum VTKSurfaceMeshFormats {
  UnknownType, VTKPolyDataType, VTKXMLPolyDataType, STLType,
  PLYType }

  Valid volume formats io functions can handle.
- enum VTKVolumeMeshFormats { UnknownVolumeType, VTKUnstructuredGridType }

  Valid volume formats io functions can handle.

Functions

- void SaveVtkShortArray (const char ∗filename, vtkShortArray ∗the_array)

  Save a vtk short array using a specific format.
- void LoadVtkShortArray (const char ∗filename, vtkShortArray ∗the_array)

  Load a vtk short array using a specific format.
- vtkPolyData ∗ GetShapeSubSurface (vtkPolyData ∗inputShape, unsigned int nSubPart)

  Call to GetShapeSubSurface( ) with nSubPart-0.1, nSubPart+0.1.
- vtkPolyData ∗ GetShapeSubSurface (vtkPolyData ∗inputShape, double tholdLower, double tholdUpper)

  Apply threshold to extract the subpart, apply vtkDataSetSurfaceFilter and vtkCleanPolyData.
- vtkPolyData ∗ CloseSurface (vtkPolyData ∗shape)

  Closes only 1 hole, make sure there are no more.
- vtkPolyData ∗ GenerateHoleCover (vtkPolyData ∗edge)

  Generate a cover for a small hole. Uses centroid.
- void SavePolydata (vtkPolyData ∗poly, const char ∗filename, bool binary=false)

  Save polydata to a file.
- void SaveImage (vtkDataSet ∗image, const char ∗filename)

  Save image to a file.
- vtkImageData ∗ LoadImage (const char ∗filename)

  Load image from a file.
- void SaveUnstructuredGrid (vtkUnstructuredGrid ∗grid, const char ∗filename)

  Save unstructured grid to a file.
- void SavePoints (vtkPoints ∗pts, const char ∗filename)

  Save vtkPoints to a file.
- bool FileExists (const char ∗filename, bool no_exception=false)

Check if file exists and throw an exception if needed.

- void ReadFilelist (const char ∗file, std::vector< std::string > &list, bool check_existence=false)

  Generate a filelist.

- vtkPolyData ∗ Points2Polydata (vtkPoints ∗points, double scalar)

  Saves points such that they can e visualized in paraview also saves a scalar corresponding to a position so it is easy to see the ordering of points (if sampling is correct) if scalars != NULL, the corresponding scalar values will be assigned to the points.

- vtkPolyData ∗ Points2Polydata (vtkPoints ∗points, const double ∗scalars=NULL)

  Saves points such that they can e visualized in paraview also saves a scalar corresponding to a position so it is easy to see the ordering of points.

- void ExportPolyDataPoints (vtkPolyData ∗shape, vnl_vector< double > &points)

  extract points from polydata

- void ExportPolyDataPoints (vtkPolyData ∗shape, vnl_matrix< double > &points)

  extract points from polydata

- void ImportPolyDataPoints (vtkPolyData ∗shape, vnl_vector< double > &points)

  copy points to polydata

- void ImportPolyDataPoints (vtkPolyData ∗shape, vnl_matrix< double > &points)

  copy points to polydata

- void GenerateDecimationScript (const char ∗filename, int nfaces)

  generate a Meshlab script for mesh decimation, specify number of faces

- void ScaleShape (vtkPolyData ∗shapein, vtkPolyData ∗shapeout, float scale, bool centerAfterScale=false)

  Rescale polydata.

- void ShrinkImage (vtkDataSet ∗imagein, vtkDataSet ∗imageout, int factor)

  Resize image.

- void ScaleVolume (vtkUnstructuredGrid ∗volumein, vtkUnstructuredGrid ∗volumeout, float scale)

  Resize unstructured grid.

- vtkPolyData ∗ LoadShapeFromFile (const char ∗shapeFileName)

  Load polydata from file.

- vtkUnstructuredGrid ∗ LoadVolumeFromFile (const char ∗volumeFileName)

  Load unstructured grid from file.

- void SaveVolumeToFile (vtkUnstructuredGrid ∗volumePt, const char ∗volumeFileName, const char ∗header)

  Save unstructured grid to file.

- void SaveShapeToFile (vtkPolyData ∗shapePt, const char ∗shapeFileName, const char ∗header=NULL)

  Save polydata to file.

- void GetP2S (vtkPolyData ∗manualPt, vtkPolyData ∗segmentedPt, double &mean, double &std_dev, double &max, double &last, bool b_array)

  Calculate point-to-surface distance.

- void GetP2P (vtkPolyData ∗manualPt, vtkPolyData ∗segmentedPt, double &mean, double &std_dev, double &max, double &last, bool b_array)

  Calculate point-to-point distance.

- vtkPolyData ∗ GetP2S (vtkPolyData ∗shapePt1, vtkPolyData ∗shapePt2, std::vector< vnl_vector< double > > &distances)

  Calculate point-to-surface distance.

- vtkPolyData ∗ GetP2S (vtkPolyData ∗shapePt1, vtkPolyData ∗shapePt2, vnl_vector< double > &distances)

  Calculate point-to-surface distance.

- void GetS2S (vtkPolyData ∗shapePt1, vtkPolyData ∗shapePt2, std::vector< vnl_vector< double > > &distances)

  Calculate surface-to-surface distance.

- bool CheckSaveFileExtension (const char ∗shapeFileName)

  check if the file has valid extension for saving. To remove one day.

- VTKVolumeMeshFormats GetTypeOfVTKVolumeData (const char ∗volumeFileName)

  identify volume data type
- VTKSurfaceMeshFormats GetTypeOfVTKData (const char ∗shapeFileName)

  identify VTK data type
- int laplace3D_voxelsize (vtkImageData ∗inputImage, vtkImageData ∗outputImage, int iterations)

  Explicit solution to Laplace Eq. (c) Ruben Cardenes + Constantine Butakoff.

4.1.1 Enumeration Type Documentation

4.1.1.1 enum CommonTools::VTKSurfaceMeshFormats

Valid volume formats io functions can handle.

Enumerator:

UnknownType

VTKPolyDataType

VTKXMLPolyDataType

STLType

PLYType

4.1.1.2 enum CommonTools::VTKVolumeMeshFormats

Valid volume formats io functions can handle.

Enumerator:

UnknownVolumeType

VTKUnstructuredGridType

4.1.2 Function Documentation

4.1.2.1 bool CommonTools::CheckSaveFileExtension (const char ∗ *shapeFileName*)

check if the file has valid extension for saving. To remove one day.

4.1.2.2 vtkPolyData ∗ CommonTools::CloseSurface (vtkPolyData ∗ *shape*)

Closes only 1 hole, make sure there are no more.

4.1.2.3 void CommonTools::ExportPolyDataPoints (vtkPolyData ∗ *shape*, vnl_vector< double > & *points*)

extract points from polydata

4.1.2.4 void CommonTools::ExportPolyDataPoints (vtkPolyData ∗ *shape*, vnl_matrix< double > & *points*)

extract points from polydata

4.1.2.5 bool CommonTools::FileExists (const char ∗ *filename*, bool *no_exception* = `false`)

Check if file exists and throw an exception if needed.

4.1.2.6 void CommonTools::GenerateDecimationScript (const char * *filename,* int *nfaces*)

generate a Meshlab script for mesh decimation, specify number of faces

4.1.2.7 vtkPolyData * CommonTools::GenerateHoleCover (vtkPolyData * *edge*)

Generate a cover for a small hole. Uses centroid.

4.1.2.8 void CommonTools::GetP2P (vtkPolyData * *manualPt,* vtkPolyData * *segmentedPt,* double & *mean,* double & *std_dev,* double & *max,* double & *last,* bool *b_array*)

Calculate point-to-point distance.

4.1.2.9 void CommonTools::GetP2S (vtkPolyData * *manualPt,* vtkPolyData * *segmentedPt,* double & *mean,* double & *std_dev,* double & *max,* double & *last,* bool *b_array*)

Calculate point-to-surface distance.

4.1.2.10 vtkPolyData * CommonTools::GetP2S (vtkPolyData * *shapePt1,* vtkPolyData * *shapePt2,* std::vector< vnl_vector< double > > & *distances*)

Calculate point-to-surface distance.

4.1.2.11 vtkPolyData * CommonTools::GetP2S (vtkPolyData * *shapePt1,* vtkPolyData * *shapePt2,* vnl_vector< double > & *distances*)

Calculate point-to-surface distance.

4.1.2.12 void CommonTools::GetS2S (vtkPolyData * *shapePt1,* vtkPolyData * *shapePt2,* std::vector< vnl_vector< double > > & *distances*)

Calculate surface-to-surface distance.

4.1.2.13 vtkPolyData * CommonTools::GetShapeSubSurface (vtkPolyData * *inputShape,* unsigned int *nSubPart*)

Call to GetShapeSubSurface( ) with nSubPart-0.1, nSubPart+0.1.

4.1.2.14 vtkPolyData * CommonTools::GetShapeSubSurface (vtkPolyData * *inputShape,* double *tholdLower,* double *tholdUpper*)

Apply threshold to extract the subpart, apply vtkDataSetSurfaceFilter and vtkCleanPolyData.

Note

> The caller to this function should delete the output shape

4.1.2.15 CommonTools::VTKSurfaceMeshFormats CommonTools::GetTypeOfVTKData (const char * *shapeFileName*)

identify VTK data type

4.1.2.16 CommonTools::VTKVolumeMeshFormats CommonTools::GetTypeOfVTKVolumeData (const char * *volumeFileName*)

identify volume data type

4.1.2.17 void CommonTools::ImportPolyDataPoints (vtkPolyData * *shape,* vnl_vector< double > & *points*)

copy points to polydata

4.1.2.18 void CommonTools::ImportPolyDataPoints (vtkPolyData * *shape,* vnl_matrix< double > & *points*)

copy points to polydata

4.1.2.19 int CommonTools::laplace3D_voxelsize (vtkImageData * *inputImage,* vtkImageData * *outputImage,* int *iterations*)

Explicit solution to Laplace Eq. (c) Ruben Cardenes + Constantine Butakoff.

Explicit solution to Laplace Eq. (c) Ruben Cardenes + Constantine Butakoff 3 Outside domain 1 Exterior boundary 0 Interior boundary 2 Inside domain

4.1.2.20 vtkImageData * CommonTools::LoadImage (const char * *filename*)

Load image from a file.

4.1.2.21 vtkPolyData * CommonTools::LoadShapeFromFile (const char * *shapeFileName*)

Load polydata from file.

4.1.2.22 vtkUnstructuredGrid * CommonTools::LoadVolumeFromFile (const char * *volumeFileName*)

Load unstructured grid from file.

4.1.2.23 void CommonTools::LoadVtkShortArray (const char * *filename,* vtkShortArray * *the_array*)

Load a vtk short array using a specific format.

Note

If there's an error, an std::exception is thrown

4.1.2.24 vtkPolyData * CommonTools::Points2Polydata (vtkPoints * *points,* double *scalar*)

Saves points such that they can e visualized in paraview also saves a scalar corresponding to a position so it is easy to see the ordering of points (if sampling is correct) if scalars != NULL, the corresponding scalar values will be assigned to the points.

Note

The caller to this function should call points->Delete( )

4.1.2.25 vtkPolyData * CommonTools::Points2Polydata (vtkPoints * *points,* const double * *scalars =* `NULL`)

Saves points such that they can e visualized in paraview also saves a scalar corresponding to a position so it is easy to see the ordering of points.

(if sampling is correct) if scalars != NULL, the corresponding scalar values will be assigned to the points

Note

The caller to this function should call points->Delete( )

4.1.2.26 void CommonTools::ReadFilelist (const char * *file,* std::vector< std::string > & *list,* bool *check_existence =* `false`)

Generate a filelist.

4.1.2.27 void CommonTools::SaveImage (vtkDataSet * *image,* const char * *filename*)

Save image to a file.

4.1.2.28 void CommonTools::SavePoints (vtkPoints * *pts,* const char * *filename*)

Save vtkPoints to a file.

4.1.2.29 void CommonTools::SavePolydata (vtkPolyData * *poly,* const char * *filename,* bool *binary =* `false`)

Save polydata to a file.

4.1.2.30 void CommonTools::SaveShapeToFile (vtkPolyData * *shapePt,* const char * *shapeFileName,* const char * *header =* `NULL`)

Save polydata to file.

4.1.2.31 void CommonTools::SaveUnstructuredGrid (vtkUnstructuredGrid * *grid,* const char * *filename*)

Save unstructured grid to a file.

4.1.2.32 void CommonTools::SaveVolumeToFile (vtkUnstructuredGrid * *volumePt,* const char * *volumeFileName,* const char * *header*)

Save unstructured grid to file.

4.1.2.33 void CommonTools::SaveVtkShortArray (const char * *filename,* vtkShortArray * *the_array*)

Save a vtk short array using a specific format.

Note

If there's an error, an std::exception is thrown

4.1.2.34 void CommonTools::ScaleShape (vtkPolyData ∗ *shapein*, vtkPolyData ∗ *shapeout*, float *scale*, bool *centerAfterScale* = false)

Rescale polydata.

4.1.2.35 void CommonTools::ScaleVolume (vtkUnstructuredGrid ∗ *volumein*, vtkUnstructuredGrid ∗ *volumeout*, float *scale*)

Resize unstructured grid.

4.1.2.36 void CommonTools::ShrinkImage (vtkDataSet ∗ *imagein*, vtkDataSet ∗ *imageout*, int *factor*)

Resize image.

4.2 itk Namespace Reference

Classes

- class BinaryThinningImageFilter3D

  This filter computes one-pixel-wide skeleton of a 3D input image.
- class ImageToVTKImageFilter

  Converts an ITK image into a VTK image and plugs a itk data pipeline to a VTK datapipeline.

Chapter 5

Class Documentation

5.1 itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage > Class Template Reference

This filter computes one-pixel-wide skeleton of a 3D input image.

```
#include <itkBinaryThinningImageFilter3D.h>
```

Public Types

- typedef BinaryThinningImageFilter3D Self
- typedef ImageToImageFilter < TInputImage, TOutputImage > Superclass
- typedef SmartPointer< Self > Pointer
- typedef SmartPointer< const Self > ConstPointer
- typedef TInputImage InputImageType
- typedef TOutputImage OutputImageType
- typedef InputImageType::RegionType RegionType
- typedef RegionType::IndexType IndexType
- typedef InputImageType::PixelType InputImagePixelType
- typedef OutputImageType::PixelType OutputImagePixelType
- typedef RegionType::SizeType SizeType
- typedef InputImageType::ConstPointer InputImagePointer
- typedef OutputImageType::Pointer OutputImagePointer
- typedef ConstantBoundaryCondition < TInputImage > ConstBoundaryConditionType
- typedef NeighborhoodIterator < TInputImage, ConstBoundaryConditionType > NeighborhoodIteratorType
- typedef NeighborhoodIteratorType::NeighborhoodType NeighborhoodType

Public Member Functions

- itkNewMacro (Self)
- itkTypeMacro (BinaryThinningImageFilter3D, ImageToImageFilter)
- OutputImageType $*$ GetThinning (void)
- itkStaticConstMacro (InputImageDimension, unsigned int, TInputImage::ImageDimension)
- itkStaticConstMacro (OutputImageDimension, unsigned int, TOutputImage::ImageDimension)

Protected Member Functions

- BinaryThinningImageFilter3D ()
- virtual ∼BinaryThinningImageFilter3D ()
- void PrintSelf (std::ostream &os, Indent indent) const
- void GenerateData ()
- void PrepareData ()
- void ComputeThinImage ()
- bool isEulerInvariant (NeighborhoodType neighbors, int ∗LUT)
- void fillEulerLUT (int ∗LUT)
- bool isSimplePoint (NeighborhoodType neighbors)
- void Octree_labeling (int octant, int label, int ∗cube)

5.1.1 Detailed Description

template<class TInputImage, class TOutputImage>class itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >

This filter computes one-pixel-wide skeleton of a 3D input image.

This class is parametrized over the type of the input image and the type of the output image.

The input is assumed to be a binary image. All non-zero valued voxels are set to 1 internally to simplify the computation. The filter will produce a skeleton of the object. The output background values are 0, and the foreground values are 1.

A 26-neighbourhood configuration is used for the foreground and a 6-neighbourhood configuration for the background. Thinning is performed symmetrically in order to guarantee that the skeleton lies medial within the object.

This filter is a parallel thinning algorithm and is an implementation of the algorithm described in:

T.C. Lee, R.L. Kashyap, and C.N. Chu. Building skeleton models via 3-D medial surface/axis thinning algorithms. Computer Vision, Graphics, and Image Processing, 56(6):462–478, 1994.

To do: Make use of multi-threading.

Author

Hanno Homann, Oxford University, Wolfson Medical Vision Lab, UK.

See Also

MorphologyImageFilter

5.1.2 Member Typedef Documentation

5.1.2.1 template<class TInputImage , class TOutputImage > typedef ConstantBoundaryCondition< TInputImage > itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::ConstBoundaryConditionType

Boundary condition type for the neighborhood iterator

5.1.2.2 template<class TInputImage , class TOutputImage > typedef SmartPointer<const Self> itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::ConstPointer

5.1.2.3 template<class TInputImage , class TOutputImage > typedef RegionType::IndexType itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::IndexType

Type for the index of the input image.

5.1.2.4 template< class TInputImage , class TOutputImage > typedef InputImageType::PixelType itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::InputImagePixelType

Type for the pixel type of the input image.

5.1.2.5 template< class TInputImage , class TOutputImage > typedef InputImageType::ConstPointer itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::InputImagePointer

Pointer Type for input image.

5.1.2.6 template< class TInputImage , class TOutputImage > typedef TInputImage itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::InputImageType

Type for input image.

5.1.2.7 template< class TInputImage , class TOutputImage > typedef NeighborhoodIterator<TInputImage, ConstBoundaryConditionType> itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::NeighborhoodIteratorType

Neighborhood iterator type

5.1.2.8 template< class TInputImage , class TOutputImage > typedef NeighborhoodIteratorType::NeighborhoodType itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::NeighborhoodType

Neighborhood type

5.1.2.9 template< class TInputImage , class TOutputImage > typedef OutputImageType::PixelType itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::OutputImagePixelType

Type for the pixel type of the input image.

5.1.2.10 template< class TInputImage , class TOutputImage > typedef OutputImageType::Pointer itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::OutputImagePointer

Pointer Type for the output image.

5.1.2.11 template< class TInputImage , class TOutputImage > typedef TOutputImage itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::OutputImageType

Type for output image: Skelenton of the object.

5.1.2.12 template< class TInputImage , class TOutputImage > typedef SmartPointer<Self> itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::Pointer

5.1.2.13 template< class TInputImage , class TOutputImage > typedef InputImageType::RegionType itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::RegionType

Type for the region of the input image.

5.1.2.14 template<class TInputImage , class TOutputImage > typedef **BinaryThinningImageFilter3D** itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::Self

Standard class typedefs.

5.1.2.15 template<class TInputImage , class TOutputImage > typedef RegionType::SizeType itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::SizeType

Type for the size of the input image.

5.1.2.16 template<class TInputImage , class TOutputImage > typedef ImageToImageFilter<TInputImage,TOutputImage> itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::Superclass

5.1.3 Constructor & Destructor Documentation

5.1.3.1 template<class TInputImage , class TOutputImage > itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::BinaryThinningImageFilter3D ( ) `[protected]`

5.1.3.2 template<class TInputImage , class TOutputImage > virtual itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::~BinaryThinningImageFilter3D ( ) `[inline]`, `[protected]`, `[virtual]`

5.1.4 Member Function Documentation

5.1.4.1 template<class TInputImage , class TOutputImage > void itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::ComputeThinImage ( ) `[protected]`

Compute thinning Image.

5.1.4.2 template<class TInputImage , class TOutputImage > void itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::fillEulerLUT (int * *LUT*) `[protected]`

5.1.4.3 template<class TInputImage , class TOutputImage > void itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::GenerateData ( ) `[protected]`

Compute thinning Image.

5.1.4.4 template<class TInputImage , class TOutputImage > OutputImageType* itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::GetThinning (void)

Get Skelenton by thinning image.

5.1.4.5 template<class TInputImage , class TOutputImage > bool itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::isEulerInvariant (NeighborhoodType *neighbors,* int * *LUT*) `[protected]`

isEulerInvariant [Lee94]

5.1.4.6 template<class TInputImage , class TOutputImage > bool itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::isSimplePoint (NeighborhoodType *neighbors*) `[protected]`

isSimplePoint [Lee94]

5.1.4.7 template<class TInputImage , class TOutputImage > itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::itkNewMacro (Self)

Method for creation through the object factory

5.1.4.8 template<class TInputImage , class TOutputImage > itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::itkStaticConstMacro (InputImageDimension , unsigned *int,* TInputImage::ImageDimension)

ImageDimension enumeration

5.1.4.9 template<class TInputImage , class TOutputImage > itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::itkStaticConstMacro (OutputImageDimension , unsigned *int,* TOutputImage::ImageDimension)

5.1.4.10 template<class TInputImage , class TOutputImage > itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::itkTypeMacro (BinaryThinningImageFilter3D< TInputImage, TOutputImage > , ImageToImageFilter)

Run-time type information (and related methods).

5.1.4.11 template<class TInputImage , class TOutputImage > void itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::Octree_labeling (int *octant,* int *label,* int * *cube*) [protected]

Octree_labeling [Lee94]

5.1.4.12 template<class TInputImage , class TOutputImage > void itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::PrepareData ( ) [protected]

Prepare data.

5.1.4.13 template<class TInputImage , class TOutputImage > void itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >::PrintSelf (std::ostream & *os,* Indent *indent*) const [protected]

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

- /home/costa/Data/Code/C/apps/imported/itkBinaryThinningImageFilter3D.h

5.2 BITMAPFILEHEADER Struct Reference

Public Attributes

- char Signature [2]
- uint32_t FileSize
- uint32_t reserved
- uint32_t DataOffset

5.2.1 Member Data Documentation

5.2.1.1 uint32_t BITMAPFILEHEADER::DataOffset

5.2.1.2 uint32_t BITMAPFILEHEADER::FileSize

5.2.1.3 uint32_t BITMAPFILEHEADER::reserved

5.2.1.4 char BITMAPFILEHEADER::Signature[2]

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

- /home/costa/Data/Code/C/apps/imported/pgm2itkvol/mcimage.c

5.3 BITMAPINFOHEADER Struct Reference

Public Attributes

- uint32_t Size
- uint32_t Width
- uint32_t Height
- uint16_t Planes
- uint16_t BitCount
- uint32_t Compression
- uint32_t ImageSize
- uint32_t XpixelsPerM
- uint32_t YpixelsPerM
- uint32_t ColorsUsed
- uint32_t ColorsImportant

5.3.1 Member Data Documentation

5.3.1.1 uint16_t BITMAPINFOHEADER::BitCount

5.3.1.2 uint32_t BITMAPINFOHEADER::ColorsImportant

5.3.1.3 uint32_t BITMAPINFOHEADER::ColorsUsed

5.3.1.4 uint32_t BITMAPINFOHEADER::Compression

5.3.1.5 uint32_t BITMAPINFOHEADER::Height

5.3.1.6 uint32_t BITMAPINFOHEADER::ImageSize

5.3.1.7 uint16_t BITMAPINFOHEADER::Planes

5.3.1.8 uint32_t BITMAPINFOHEADER::Size

5.3.1.9 uint32_t BITMAPINFOHEADER::Width

5.3.1.10 uint32_t BITMAPINFOHEADER::XpixelsPerM

5.3.1.11 uint32_t BITMAPINFOHEADER::YpixelsPerM

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

- /home/costa/Data/Code/C/apps/imported/pgm2itkvol/mcimage.c

5.4 dcomplex Struct Reference

Complex number, represented by doubles.

```
#include <mccodimage.h>
```

Public Attributes

- double re

  real part
- double im

  imaginary part

5.4.1 Detailed Description

Complex number, represented by doubles.

5.4.2 Member Data Documentation

5.4.2.1 double dcomplex::im

imaginary part

5.4.2.2 double dcomplex::re

real part

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

- /home/costa/Data/Code/C/apps/imported/pgm2itkvol/mccodimage.h

5.5 fcomplex Struct Reference

Complex number, represented by floats.

```
#include <mccodimage.h>
```

Public Attributes

- float re

  real part
- float im

  imaginary part

5.5.1 Detailed Description

Complex number, represented by floats.

Note

'fcomplex' is necessary because of msvc

5.5.2 Member Data Documentation

5.5.2.1 float fcomplex::im

imaginary part

5.5.2.2 float fcomplex::re

real part

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

- /home/costa/Data/Code/C/apps/imported/pgm2itkvol/mccodimage.h

5.6 itk::ImageToVTKImageFilter< TInputImage > Class Template Reference

Converts an ITK image into a VTK image and plugs a itk data pipeline to a VTK datapipeline.

```
#include <itkImageToVTKImageFilter.h>
```

Public Types

- typedef ImageToVTKImageFilter Self
- typedef ProcessObject Superclass
- typedef SmartPointer< Self > Pointer
- typedef SmartPointer< const Self > ConstPointer
- typedef TInputImage InputImageType
- typedef InputImageType::ConstPointer InputImagePointer
- typedef VTKImageExport < InputImageType > ExporterFilterType
- typedef ExporterFilterType::Pointer ExporterFilterPointer

Public Member Functions

- itkNewMacro (Self)
- itkTypeMacro (ImageToVTKImageFilter, ProcessObject)
- vtkImageData ∗ GetOutput () const
- void SetInput (const InputImageType ∗)
- vtkImageImport ∗ GetImporter () const
- ExporterFilterType ∗ GetExporter () const
- void Update ()
- const std::vector< double > & getvtest () const
- int testsize ()
- std::vector< double > addvector (const std::vector< double > &v)
- const std::vector< double > & addtest (double toto)
- std::vector< double > tralala ()

Protected Member Functions

- ImageToVTKImageFilter ()
- virtual ∼ImageToVTKImageFilter ()

5.6.1 Detailed Description

template<class TInputImage>class itk::ImageToVTKImageFilter< TInputImage >

Converts an ITK image into a VTK image and plugs a itk data pipeline to a VTK datapipeline.

This class puts together an itkVTKImageExporter and a vtkImageImporter. It takes care of the details related to the connection of ITK and VTK pipelines. The User will perceive this filter as an adaptor to which an itk::Image can be plugged as input and a vtkImage is produced as output.

5.6.2 Member Typedef Documentation

5.6.2.1 template<class TInputImage > typedef SmartPointer<const Self> itk::ImageToVTKImageFilter< TInputImage >::ConstPointer

5.6.2.2 template<class TInputImage > typedef ExporterFilterType::Pointer itk::ImageToVTKImageFilter< TInputImage >::ExporterFilterPointer

5.6.2.3 template<class TInputImage > typedef VTKImageExport< InputImageType> itk::ImageToVTKImageFilter< TInputImage >::ExporterFilterType

5.6.2.4 template<class TInputImage > typedef InputImageType::ConstPointer itk::ImageToVTKImageFilter< TInputImage >::InputImagePointer

5.6.2.5 template<class TInputImage > typedef TInputImage itk::ImageToVTKImageFilter< TInputImage >::InputImageType

Some typedefs.

5.6.2.6 template<class TInputImage > typedef SmartPointer<Self> itk::ImageToVTKImageFilter< TInputImage >::Pointer

5.6.2.7 template<class TInputImage > typedef ImageToVTKImageFilter itk::ImageToVTKImageFilter< TInputImage >::Self

Standard class typedefs.

5.6.2.8 template<class TInputImage > typedef ProcessObject itk::ImageToVTKImageFilter< TInputImage >::Superclass

5.6.3 Constructor & Destructor Documentation

5.6.3.1 template<class TInputImage > itk::ImageToVTKImageFilter< TInputImage >::ImageToVTKImageFilter ( ) `[protected]`

5.6.3.2 template<class TInputImage > virtual itk::ImageToVTKImageFilter< TInputImage >::~ImageToVTKImageFilter ( ) `[protected]`, `[virtual]`

5.6.4 Member Function Documentation

5.6.4.1 template<class TInputImage > const std::vector<double>& itk::ImageToVTKImageFilter< TInputImage >::addtest (double *toto*) `[inline]`

5.6.4.2 template< class TInputImage > std::vector<double> itk::ImageToVTKImageFilter< TInputImage >::addvector (const std::vector< double > & *v*) [inline]

5.6.4.3 template< class TInputImage > ExporterFilterType∗ itk::ImageToVTKImageFilter< TInputImage >::GetExporter ( ) const

Return the internal ITK image exporter filter. This is intended to facilitate users the access to methods in the exporter

5.6.4.4 template< class TInputImage > vtkImageImport∗ itk::ImageToVTKImageFilter< TInputImage >::GetImporter ( ) const

Return the internal VTK image importer filter. This is intended to facilitate users the access to methods in the importer

5.6.4.5 template< class TInputImage > vtkImageData∗ itk::ImageToVTKImageFilter< TInputImage >::GetOutput ( ) const

Get the output in the form of a vtkImage. This call is delegated to the internal vtkImageImporter filter

5.6.4.6 template< class TInputImage > const std::vector<double>& itk::ImageToVTKImageFilter< TInputImage >::getvtest ( ) const [inline]

5.6.4.7 template< class TInputImage > itk::ImageToVTKImageFilter< TInputImage >::itkNewMacro (Self)

Method for creation through the object factory.

5.6.4.8 template< class TInputImage > itk::ImageToVTKImageFilter< TInputImage >::itkTypeMacro (ImageToVTKImageFilter< TInputImage > , ProcessObject)

Run-time type information (and related methods).

5.6.4.9 template< class TInputImage > void itk::ImageToVTKImageFilter< TInputImage >::SetInput (const InputImageType ∗)

Set the input in the form of an itk::Image

5.6.4.10 template< class TInputImage > int itk::ImageToVTKImageFilter< TInputImage >::testsize ( ) [inline]

5.6.4.11 template< class TInputImage > std::vector<double> itk::ImageToVTKImageFilter< TInputImage >::tralala ( ) [inline]

5.6.4.12 template< class TInputImage > void itk::ImageToVTKImageFilter< TInputImage >::Update ( )

This call delegate the update to the importer

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

- /home/costa/Data/Code/C/apps/imported/itkImageToVTKImageFilter.h

5.7 RGBFILEHEADER Struct Reference

Public Attributes

- uint16_t magic
- uint8_t compression
- uint8_t bytespercha
- uint16_t dim
- uint16_t width
- uint16_t height
- uint16_t components
- uint32_t mincol
- uint32_t maxcol
- uint32_t dummy
- char name [80]
- uint32_t cmaptype

5.7.1 Member Data Documentation

5.7.1.1 uint8_t RGBFILEHEADER::bytespercha

5.7.1.2 uint32_t RGBFILEHEADER::cmaptype

5.7.1.3 uint16_t RGBFILEHEADER::components

5.7.1.4 uint8_t RGBFILEHEADER::compression

5.7.1.5 uint16_t RGBFILEHEADER::dim

5.7.1.6 uint32_t RGBFILEHEADER::dummy

5.7.1.7 uint16_t RGBFILEHEADER::height

5.7.1.8 uint16_t RGBFILEHEADER::magic

5.7.1.9 uint32_t RGBFILEHEADER::maxcol

5.7.1.10 uint32_t RGBFILEHEADER::mincol

5.7.1.11 char RGBFILEHEADER::name[80]

5.7.1.12 uint16_t RGBFILEHEADER::width

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

- /home/costa/Data/Code/C/apps/imported/pgm2itkvol/mcimage.c

5.8 sorting_struct Struct Reference

Public Attributes

- double value
- int index

5.8.1 Member Data Documentation

5.8.1.1 int sorting_struct::index

5.8.1.2 double sorting_struct::value

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

- /home/costa/Data/Code/C/apps/MRIRemesh.cpp

5.9 xvimage Struct Reference

The image class for the C functions.

```
#include <mccodimage.h>
```

Public Attributes

- char ∗ name

  Dummy - not used anymore.
- index_t row_size

  Size of a row (number of columns)
- index_t col_size

  Size of a column (number of rows)
- index_t depth_size

  Number of planes (for 3d images)
- index_t time_size

  Number of (2d or 3d) images.
- index_t num_data_bands

  Number of bands per data pixel, or number of bands per image, or dimension of vector data, or number of elements in a vector.
- index_t d

  The dimension of the image.
- int32_t data_storage_type

  Storage type for disk data.
- double xdim

  Voxel dimensions in real world.
- double ydim
- double zdim
- double origin_x

  Origin in real world.
- double origin_y
- double origin_z
- index_t xmin

  Region of interest: x coordinates.
- index_t xmax
- index_t ymin

  Region of interest: y coordinates.
- index_t ymax
- index_t zmin

  Region of interest: z coordinates.
- index_t zmax
- void ∗ image_data

  Pointer on raw data.

5.9.1 Detailed Description

The image class for the C functions.

This class holds the image data for the C functions of Pink.

5.9.2 Member Data Documentation

5.9.2.1 index_t xvimage::col_size

Size of a column (number of rows)

5.9.2.2 index_t xvimage::d

The dimension of the image.

5.9.2.3 int32_t xvimage::data_storage_type

Storage type for disk data.

5.9.2.4 index_t xvimage::depth_size

Number of planes (for 3d images)

5.9.2.5 void∗ xvimage::image_data

Pointer on raw data.

5.9.2.6 char∗ xvimage::name

Dummy - not used anymore.

5.9.2.7 index_t xvimage::num_data_bands

Number of bands per data pixel, or number of bands per image, or dimension of vector data, or number of elements in a vector.

5.9.2.8 double xvimage::origin_x

Origin in real world.

5.9.2.9 double xvimage::origin_y

5.9.2.10 double xvimage::origin_z

5.9.2.11 index_t xvimage::row_size

Size of a row (number of columns)

5.9.2.12 index_t xvimage::time_size

Number of (2d or 3d) images.

5.9.2.13 double xvimage::xdim

Voxel dimensions in real world.

5.9.2.14 index_t xvimage::xmax

5.9.2.15 index_t xvimage::xmin

Region of interest: x coordinates.

5.9.2.16 double xvimage::ydim

5.9.2.17 index_t xvimage::ymax

5.9.2.18 index_t xvimage::ymin

Region of interest: y coordinates.

5.9.2.19 double xvimage::zdim

5.9.2.20 index_t xvimage::zmax

5.9.2.21 index_t xvimage::zmin

Region of interest: z coordinates.

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

- /home/costa/Data/Code/C/apps/imported/pgm2itkvol/mccodimage.h

Chapter 6

File Documentation

6.1 /home/costa/Data/Code/C/apps/Blob.cpp File Reference

Active surfaces. An active surfaces implementation that fits a 3D blob to a set of points. Under construction.

```
#include <vtkSmartPointer.h>
#include "CommonTools.h"
#include <vtkPolyData.h>
#include <vtkPolyDataNormals.h>
#include <vtkDoubleArray.h>
#include <vtkMassProperties.h>
#include <vtkCurvatures.h>
#include <vtkPointData.h>
#include <vtkSphereSource.h>
#include <vtkDataSetSurfaceFilter.h>
#include <vtkPoints.h>
#include <vtkPointLocator.h>
#include <vtkTriangle.h>
#include <vtkCellData.h>
#include <vtkCellArray.h>
#include <vtkType.h>
#include <vector>
#include <iostream>
```

Functions

- void ApproximateCurvature (vtkPolyData ∗mesh, vtkDoubleArray ∗normals)

    A function that aproximates curvature.

- void GetPointNeighbors (vtkPolyData ∗mesh, vtkIdType ptid, vtkIdList ∗ptIds)

    A function that gets point's neighbors.

- int main (int argc, char ∗∗argv)

6.1.1 Detailed Description

Active surfaces. An active surfaces implementation that fits a 3D blob to a set of points. Under construction.

6.1.2 Function Documentation

6.1.2.1 void ApproximateCurvature (vtkPolyData ∗ *mesh,* vtkDoubleArray ∗ *normals*)

A function that aproximates curvature.

6.1.2.2 void GetPointNeighbors (vtkPolyData ∗ *mesh,* vtkIdType *ptid,* vtkIdList ∗ *ptIds*)

A function that gets point's neighbors.

6.1.2.3 int main (int *argc,* char ∗∗ *argv*)

6.2 /home/costa/Data/Code/C/apps/chaste2vtk.cpp File Reference

Convert chaste .ele .node tetrahedral mesh to VTK unstructured grid (only for Oxford Rabbit)

```
#include <vtkSmartPointer.h>
#include <vtkUnstructuredGridWriter.h>
#include <vtkUnstructuredGrid.h>
#include <vtkType.h>
#include <vtkTetra.h>
#include <vtkIntArray.h>
#include <vtkCellData.h>
#include <vtkCellArray.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <vnl/vnl_vector.h>
#include "CommonTools.h"
```

Functions

- void ReadChasteNodes (const char ∗filename, vtkPoints ∗pts)

  Reads binary chaste .node file and stores into vtkPoints.
- void ReadChasteElements (const char ∗filename, vtkCellArray ∗cells, vtkIntArray ∗scalars)

  Reads binary chaste .ele file of tetrahedra and stores into vtkCells, it also retrieves the first scalar array.
- int main (int argc, char ∗∗argv)

6.2.1 Detailed Description

Convert chaste .ele .node tetrahedral mesh to VTK unstructured grid (only for Oxford Rabbit)

6.2.2 Function Documentation

6.2.2.1 int main (int *argc,* char ∗∗ *argv*)

6.2.2.2 void ReadChasteElements (const char ∗ *filename,* vtkCellArray ∗ *cells,* vtkIntArray ∗ *scalars*)

Reads binary chaste .ele file of tetrahedra and stores into vtkCells, it also retrieves the first scalar array.

6.2.2.3 void ReadChasteNodes (const char ∗ *filename,* vtkPoints ∗ *pts*)

Reads binary chaste .node file and stores into vtkPoints.

6.3 /home/costa/Data/Code/C/apps/CloseBVMesh.cpp File Reference

Closes biventricular mesh by connecting endocardial edge to epicardial at the base. Does not close ventricles.

```
#include "CommonTools.h"
#include <vtkSmartPointer.h>
#include <vtkPolyData.h>
#include <vtkPolyDataConnectivityFilter.h>
#include <vtkFeatureEdges.h>
#include <vtkDelaunay2D.h>
#include <vtkCell.h>
#include <vtkPointData.h>
#include <vtkCellArray.h>
#include <vtkShortArray.h>
#include <vtkCellData.h>
#include <vtkAppendPolyData.h>
#include <vtkCleanPolyData.h>
#include <vtkCellLocator.h>
#include <vtkPolygon.h>
#include <vtkStripper.h>
```

Functions

- void CopyCellScalars (vtkPolyData ∗src, vtkPolyData ∗tgt, char ∗scalars_name, int fill_value)

    Copies cell scalars from source mesh to the target.
- vtkPolygon ∗ PolyData2Polygon (vtkPolyData ∗pd)

    Convert vtkPolyData polygon to vtkPolygon.
- int PointInPolygon (double x[3], vtkPolygon ∗pg)

    Verify if a point is in vtkPolygon.
- double PolygonBoundaryArea (vtkPolygon ∗polygon)

    Calculate area of a vtkPolygon.
- vtkPolyData ∗ FillSmallHoles (vtkPolyData ∗pd)

    Fills small holes in vtkPolyData by connecting vertices to the centroid.
- int main (int argc, char ∗argv[ ])

6.3.1 Detailed Description

Closes biventricular mesh by connecting endocardial edge to epicardial at the base. Does not close ventricles. It was made for Rafa's biventricular model. The mesh must have epicardium, rv endo and lv endo separable. No scalars are necessary. The scalars must be vtkShortArray!!! (type short)

6.3.2 Function Documentation

6.3.2.1 void CopyCellScalars (vtkPolyData ∗ *src,* vtkPolyData ∗ *tgt,* char ∗ *scalars_name,* int *fill_value*)

Copies cell scalars from source mesh to the target.

6.3.2.2 vtkPolyData ∗ FillSmallHoles (vtkPolyData ∗ *pd*)

Fills small holes in vtkPolyData by connecting vertices to the centroid.

6.3.2.3 int main (int *argc*, char ∗ *argv[ ]*)

6.3.2.4 int PointInPolygon (double *x[3]*, vtkPolygon ∗ *pg*)

Verify if a point is in vtkPolygon.

6.3.2.5 vtkPolygon ∗ PolyData2Polygon (vtkPolyData ∗ *pd*)

Convert vtkPolyData polygon to vtkPolygon.

6.3.2.6 double PolygonBoundaryArea (vtkPolygon ∗ *polygon*)

Calculate area of a vtkPolygon.

6.4 /home/costa/Data/Code/C/apps/CommonTools.cpp File Reference

```
#include <vtkCleanPolyData.h>
#include <vtkThreshold.h>
#include <vtkDataSetSurfaceFilter.h>
#include <vtkPolyDataWriter.h>
#include <vtkTriangleFilter.h>
#include <vtkAppendPolyData.h>
#include <vtkDelaunay2D.h>
#include <vtkCellArray.h>
#include <vtkFloatArray.h>
#include <vtkPointData.h>
#include <vtkCellData.h>
#include <vtkDataSetWriter.h>
#include <vtkTransform.h>
#include <vtkTransformPolyDataFilter.h>
#include <vtkImageGaussianSmooth.h>
#include <vtkImageShrink3D.h>
#include <vtkImageData.h>
#include <vtkShortArray.h>
#include <vtkPolyData.h>
#include <vtkDataSet.h>
#include <vtkPoints.h>
#include <vtkStripper.h>
#include <vtkCutter.h>
#include <vtkPointLocator.h>
#include <vtkPlane.h>
#include <vtkStringArray.h>
#include <vtkSmartPointer.h>
#include <vtkDataArray.h>
#include <vtkDataSetReader.h>
#include <vtkFeatureEdges.h>
#include <vtkTransformFilter.h>
#include <vtkUnstructuredGrid.h>
#include <vtkPolyDataConnectivityFilter.h>
#include <vtkPLYReader.h>
#include <vtkSTLReader.h>
#include <vtkPolyDataReader.h>
#include <vtkXMLPolyDataReader.h>
#include <vtkUnstructuredGridReader.h>
#include <itksys/SystemTools.hxx>
#include <vtkUnstructuredGridWriter.h>
#include <vtkPLYWriter.h>
#include <vtkSTLWriter.h>
#include <vtkXMLPolyDataWriter.h>
#include <vtkIVWriter.h>
#include <vtkCellLocator.h>
#include "CommonTools.h"
#include <stdio.h>
#include <stdlib.h>
#include <fstream>
#include <stdexcept>
#include <vnl/vnl_matrix.h>
```

6.5 /home/costa/Data/Code/C/apps/CommonTools.h File Reference

Some common functions.

```
#include <vector>
#include <vnl/vnl_vector.h>
#include <vtkImageData.h>
```

Namespaces

- namespace CommonTools

Enumerations

- enum CommonTools::VTKSurfaceMeshFormats {
CommonTools::UnknownType, CommonTools::VTKPolyDataType, CommonTools::VTKXMLPolyDataType,
CommonTools::STLType,
CommonTools::PLYType }

  Valid volume formats io functions can handle.
- enum CommonTools::VTKVolumeMeshFormats { CommonTools::UnknownVolumeType, CommonTools::VTKUnstructuredGridType }

  Valid volume formats io functions can handle.

Functions

- void CommonTools::SaveVtkShortArray (const char ∗filename, vtkShortArray ∗the_array)

  Save a vtk short array using a specific format.
- void CommonTools::LoadVtkShortArray (const char ∗filename, vtkShortArray ∗the_array)

  Load a vtk short array using a specific format.
- vtkPolyData ∗ CommonTools::GetShapeSubSurface (vtkPolyData ∗inputShape, unsigned int nSubPart)

  Call to GetShapeSubSurface( ) with nSubPart-0.1, nSubPart+0.1.
- vtkPolyData ∗ CommonTools::GetShapeSubSurface (vtkPolyData ∗inputShape, double tholdLower, double tholdUpper)

  Apply threshold to extract the subpart, apply vtkDataSetSurfaceFilter and vtkCleanPolyData.
- vtkPolyData ∗ CommonTools::CloseSurface (vtkPolyData ∗shape)

  Closes only 1 hole, make sure there are no more.
- vtkPolyData ∗ CommonTools::GenerateHoleCover (vtkPolyData ∗edge)

  Generate a cover for a small hole. Uses centroid.
- void CommonTools::SavePolydata (vtkPolyData ∗poly, const char ∗filename, bool binary=false)

  Save polydata to a file.
- void CommonTools::SaveImage (vtkDataSet ∗image, const char ∗filename)

  Save image to a file.
- vtkImageData ∗ CommonTools::LoadImage (const char ∗filename)

  Load image from a file.
- void CommonTools::SaveUnstructuredGrid (vtkUnstructuredGrid ∗grid, const char ∗filename)

  Save unstructured grid to a file.
- void CommonTools::SavePoints (vtkPoints ∗pts, const char ∗filename)

  Save vtkPoints to a file.
- bool CommonTools::FileExists (const char ∗filename, bool no_exception=false)

  Check if file exists and throw an exception if needed.

- void CommonTools::ReadFilelist (const char ∗file, std::vector< std::string > &list, bool check_existence=false)

    Generate a filelist.
- vtkPolyData ∗ CommonTools::Points2Polydata (vtkPoints ∗points, double scalar)

    Saves points such that they can e visualized in paraview also saves a scalar corresponding to a position so it is easy to see the ordering of points (if sampling is correct) if scalars != NULL, the corresponding scalar values will be assigned to the points.
- vtkPolyData ∗ CommonTools::Points2Polydata (vtkPoints ∗points, const double ∗scalars=NULL)

    Saves points such that they can e visualized in paraview also saves a scalar corresponding to a position so it is easy to see the ordering of points.
- void CommonTools::ExportPolyDataPoints (vtkPolyData ∗shape, vnl_vector< double > &points)

    extract points from polydata
- void CommonTools::ExportPolyDataPoints (vtkPolyData ∗shape, vnl_matrix< double > &points)

    extract points from polydata
- void CommonTools::ImportPolyDataPoints (vtkPolyData ∗shape, vnl_vector< double > &points)

    copy points to polydata
- void CommonTools::ImportPolyDataPoints (vtkPolyData ∗shape, vnl_matrix< double > &points)

    copy points to polydata
- void CommonTools::GenerateDecimationScript (const char ∗filename, int nfaces)

    generate a Meshlab script for mesh decimation, specify number of faces
- void CommonTools::ScaleShape (vtkPolyData ∗shapein, vtkPolyData ∗shapeout, float scale, bool centerAfterScale=false)

    Rescale polydata.
- void CommonTools::ShrinkImage (vtkDataSet ∗imagein, vtkDataSet ∗imageout, int factor)

    Resize image.
- void CommonTools::ScaleVolume (vtkUnstructuredGrid ∗volumein, vtkUnstructuredGrid ∗volumeout, float scale)

    Resize unstructured grid.
- vtkPolyData ∗ CommonTools::LoadShapeFromFile (const char ∗shapeFileName)

    Load polydata from file.
- vtkUnstructuredGrid ∗ CommonTools::LoadVolumeFromFile (const char ∗volumeFileName)

    Load unstructured grid from file.
- void CommonTools::SaveVolumeToFile (vtkUnstructuredGrid ∗volumePt, const char ∗volumeFileName, const char ∗header)

    Save unstructured grid to file.
- void CommonTools::SaveShapeToFile (vtkPolyData ∗shapePt, const char ∗shapeFileName, const char ∗header=NULL)

    Save polydata to file.
- void CommonTools::GetP2S (vtkPolyData ∗manualPt, vtkPolyData ∗segmentedPt, double &mean, double &std_dev, double &max, double &last, bool b_array)

    Calculate point-to-surface distance.
- void CommonTools::GetP2P (vtkPolyData ∗manualPt, vtkPolyData ∗segmentedPt, double &mean, double &std_dev, double &max, double &last, bool b_array)

    Calculate point-to-point distance.
- vtkPolyData ∗ CommonTools::GetP2S (vtkPolyData ∗shapePt1, vtkPolyData ∗shapePt2, std::vector< vnl_vector< double > > &distances)

    Calculate point-to-surface distance.
- vtkPolyData ∗ CommonTools::GetP2S (vtkPolyData ∗shapePt1, vtkPolyData ∗shapePt2, vnl_vector< double > &distances)

    Calculate point-to-surface distance.
- void CommonTools::GetS2S (vtkPolyData ∗shapePt1, vtkPolyData ∗shapePt2, std::vector< vnl_vector< double > > &distances)

Calculate surface-to-surface distance.

- bool CommonTools::CheckSaveFileExtension (const char ∗shapeFileName)

  check if the file has valid extension for saving. To remove one day.
- VTKVolumeMeshFormats CommonTools::GetTypeOfVTKVolumeData (const char ∗volumeFileName)

  identify volume data type
- VTKSurfaceMeshFormats CommonTools::GetTypeOfVTKData (const char ∗shapeFileName)

  identify VTK data type
- int CommonTools::laplace3D_voxelsize (vtkImageData ∗inputImage, vtkImageData ∗outputImage, int iterations)

  Explicit solution to Laplace Eq. (c) Ruben Cardenes + Constantine Butakoff.

6.5.1 Detailed Description

Some common functions.

6.6 /home/costa/Data/Code/C/apps/CreateImageMask.cpp File Reference

Create mask for a given shape with user defined dimensions.

```
#include <vtkPolyData.h>
#include <vtkPolyDataToImageStencil.h>
#include <vtkImageStencil.h>
#include <vtkImageData.h>
#include <vtkDataSet.h>
#include <vtkDataSetReader.h>
#include <vtkDataSetWriter.h>
#include <vtkSmartPointer.h>
#include "vtkTransform.h"
#include "vtkTransformPolyDataFilter.h"
#include "CommonTools.h"
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
```

Functions

- int main (int argc, char ∗∗argv)

6.6.1 Detailed Description

Create mask for a given shape with user defined dimensions.

6.6.2 Function Documentation

6.6.2.1 int main (int *argc,* char ∗∗ *argv*)

6.7 /home/costa/Data/Code/C/apps/ExtractShapeRegion.cpp File Reference

Extract a part of polydata.

```
#include <vtkCell.h>
#include <vtkIdList.h>
#include <vtkCleanPolyData.h>
#include <vtkPointData.h>
#include <vtkShortArray.h>
#include <vtkPolyData.h>
#include <vtkSTLReader.h>
#include <vtkXMLPolyDataReader.h>
#include <vtkPolyDataWriter.h>
#include <vtkPolyDataReader.h>
#include <vtkObject.h>
#include "CommonTools.h"
#include <vtkstd/exception>
#include <vtkFeatureEdges.h>
#include <vnl/vnl_matrix.h>
#include <vnl/vnl_vector.h>
#include <vnl/algo/vnl_svd_economy.h>
#include <vtkXMLPolyDataWriter.h>
#include <vtkPoints.h>
#include <vtkCellArray.h>
#include <vtkAppendPolyData.h>
```

Functions

- int main (int argc, char ∗argv[])

6.7.1 Detailed Description

Extract a part of polydata.

6.7.2 Function Documentation

6.7.2.1 int main (int *argc,* char ∗ *argv[]*)

6.8 /home/costa/Data/Code/C/apps/GenerateVolumetricLVMesh.cpp File Reference

Generate volumetric mesh of Left Ventricle.

```
#include <vtkCleanPolyData.h>
#include <vtkPolyData.h>
#include <vtkPolyDataReader.h>
#include <vtkIVWriter.h>
#include <vtkFloatArray.h>
#include <vtkPointData.h>
#include <vtkLookupTable.h>
#include "CommonTools.h"
#include "vtkSmartPointer.h"
#include <vector>
#include "vtkCellLocator.h"
#include "vtkPolyDataNormals.h"
#include "vnl/vnl_vector.h"
#include "vnl/vnl_matrix.h"
#include "vnl/vnl_cross.h"
#include "vtkGenericCell.h"
#include "vtkWedge.h"
#include "vtkIdList.h"
#include "vtkCell.h"
#include "vtkUnstructuredGrid.h"
#include "vtkUnstructuredGridWriter.h"
#include "vtkCellType.h"
#include "vtkCellData.h"
#include "vtkImageEuclideanDistance.h"
#include "vtkPolyDataToImageStencil.h"
#include "vtkImageStencil.h"
#include "vtkImageGradient.h"
#include "vtkImageContinuousErode3D.h"
#include "vtkImageContinuousDilate3D.h"
#include "vtkShortArray.h"
#include "vtkCellArray.h"
#include "vtkStreamTracer.h"
#include "vtkImageMathematics.h"
#include "vtkImageCast.h"
#include "vtkAssignAttribute.h"
#include "vtkSplineFilter.h"
```

Macros

- #define FIELD_DT 0
- #define FIELD_LAPLACE 1

Functions

- void GenerateLayersAlongNormals (vtkPoints ∗layers, vtkPolyData ∗epi, vtkPolyData ∗endo, int nLayers)

  generate points between epi and endo along surface normals
- void GenerateLayersAlongField (vtkPoints ∗layers, vtkPolyData ∗epi, vtkPolyData ∗endo, int nLayers, int field_type=0, float VoxelSize=0.5, int laplace_iterations=100)

  generate points between epi and endo along vector field
- void GenerateImageMask (vtkImageData ∗res_image, float fg, float bg, vtkPolyData ∗endo, vtkPolyData ∗epi, float VoxelSize=0.5)

  generate mask given epi and endo
- void GenerateLocalCoordinateCircLongit (vtkUnstructuredGrid ∗volmesh, int nLayers, int nPointsPerLevel)

  generate local coordinates

- void GenerateLocalCoordinateRadial (vtkUnstructuredGrid ∗volmesh, int nLayers)

  generate local coordinates
- void usage (char ∗exe)
- int main (int argc, char ∗∗argv)

6.8.1 Detailed Description

Generate volumetric mesh of Left Ventricle.

6.8.2 Macro Definition Documentation

6.8.2.1 #define FIELD_DT 0

6.8.2.2 #define FIELD_LAPLACE 1

6.8.3 Function Documentation

6.8.3.1 void GenerateImageMask (vtkImageData ∗ *res_image,* float *fg,* float *bg,* vtkPolyData ∗ *endo,* vtkPolyData ∗ *epi,* float *VoxelSize* = `0.5`)

generate mask given epi and endo

6.8.3.2 void GenerateLayersAlongField (vtkPoints ∗ *layers,* vtkPolyData ∗ *epi,* vtkPolyData ∗ *endo,* int *nLayers,* int *field_type* = `0`, float *VoxelSize* = `0.5`, int *laplace_iterations* = `100`)

generate points between epi and endo along vector field

6.8.3.3 void GenerateLayersAlongNormals (vtkPoints ∗ *layers,* vtkPolyData ∗ *epi,* vtkPolyData ∗ *endo,* int *nLayers*)

generate points between epi and endo along surface normals

6.8.3.4 void GenerateLocalCoordinateCircLongit (vtkUnstructuredGrid ∗ *volmesh,* int *nLayers,* int *nPointsPerLevel*)

generate local coordinates

6.8.3.5 void GenerateLocalCoordinateRadial (vtkUnstructuredGrid ∗ *volmesh,* int *nLayers*)

generate local coordinates

6.8.3.6 int main (int *argc,* char ∗∗ *argv*)

6.8.3.7 void usage (char ∗ *exe*)

6.9 /home/costa/Data/Code/C/apps/imported/itkBinaryThinningImageFilter3D.h File Reference

```
#include <itkNeighborhoodIterator.h>
#include <itkImageToImageFilter.h>
#include <itkImageRegionIteratorWithIndex.h>
#include <itkConstantBoundaryCondition.h>
#include "itkBinaryThinningImageFilter3D.txx"
```

Classes

- class itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >

  This filter computes one-pixel-wide skeleton of a 3D input image.

Namespaces

- namespace itk

6.10 /home/costa/Data/Code/C/apps/imported/itkImageToVTKImageFilter.h File Reference

```
#include "itkVTKImageExport.h"
#include "vtkImageImport.h"
#include "vtkImageData.h"
#include <vector>
#include "itkImageToVTKImageFilter.txx"
```

Classes

- class itk::ImageToVTKImageFilter< TInputImage >

  Converts an ITK image into a VTK image and plugs a itk data pipeline to a VTK datapipeline.

Namespaces

- namespace itk

6.11 /home/costa/Data/Code/C/apps/imported/LabelBranches3D.cpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <vector>
#include <iostream>
#include <sys/time.h>
#include <itkImage.h>
#include <itkImageFileReader.h>
#include <itkImageFileWriter.h>
#include <itkImageRegionIterator.h>
```

Macros

- #define FLT_MAX 3.40282347e+38F;

Functions

- void print_timing (FILE ∗fp, struct timeval start, struct timeval end)
- int mapIndex3D (int r, int c, int z, int nr, int nc, int nz)
- int maptox (int mapindex, int max1, int max2)
- int maptoy (int mapindex, int max1, int max2)
- int maptoz (int mapindex, int max1, int max2)
- void reserve_memory_triple_float (float ∗∗∗out, int max1, int max2, int max3)
- void reserve_memory_triple_int (int ∗∗∗out, int max1, int max2, int max3)
- int find_maximun (float ∗a, int num)
- int findmaximum_centerline (float ∗maps, std::vector< int > index_esqueleto)
- void CheckConnect (unsigned char ∗∗∗input, unsigned char ∗output, int max1, int max2, int max3)
- float distance (int mapindex1, int mapindex2, int max1, int max2)
- int check_neighborhood (int index_a, int index_b, int max1, int max2)
- void LabelBranchsNew (unsigned short ∗vol_esqueleto, int max1, int max2, int max3)
- void LabelBranchs (unsigned short ∗vol_esqueleto, int max1, int max2, int max3)
- int check_num_connected_neighbors (std::vector< int > neigh, int max1, int max2)
- int CheckPointState (unsigned short ∗vol_esqueleto, int mapindex, int max1, int max2, int max3, int &index_pto_triple_aux)
- void FindTriplePoints (std::vector< int > index_esqueleto, unsigned short ∗vol_esqueleto, std::vector< int > &ptos_extremos, int max1, int max2, int max3)
- int comp_conexas (std::vector< int > index_esqueleto, unsigned short ∗vol_esqueleto, int max1, int max2, int max3)
- int find_shorter_branch (std::vector< int > &vec_index_esq, unsigned short ∗output, int num_comp)
- void Relabeling (unsigned short ∗output, int max1, int max2, int max3)
- int main (int argc, char ∗argv[])

Variables

- int verbose
- int debug

6.11.1 Macro Definition Documentation

6.11.1.1 #define FLT_MAX 3.40282347e+38F;

6.11.2 Function Documentation

6.11.2.1 int check_neighborhood (int *index_a,* int *index_b,* int *max1,* int *max2*)

6.11.2.2 int check_num_connected_neighbors (std::vector< int > *neigh,* int *max1,* int *max2*)

6.11.2.3 void CheckConnect (unsigned char ∗∗∗ *input,* unsigned char ∗ *output,* int *max1,* int *max2,* int *max3*)

6.11.2.4 int CheckPointState (unsigned short ∗ *vol_esqueleto,* int *mapindex,* int *max1,* int *max2,* int *max3,* int & *index_pto_triple_aux*)

6.11.2.5 int comp_conexas (std::vector< int > *index_esqueleto,* unsigned short ∗ *vol_esqueleto,* int *max1,* int *max2,* int *max3*)

6.11.2.6 float distance (int *mapindex1,* int *mapindex2,* int *max1,* int *max2*)

6.11.2.7 int find_maximun (float ∗ *a,* int *num*)

6.11.2.8 int find_shorter_branch (std::vector< int > & *vec_index_esq,* unsigned short ∗ *output,* int *num_comp*)

6.11.2.9 int findmaximum_centerline (float ∗ *maps,* std::vector< int > *index_esqueleto*)

6.11.2.10 void FindTriplePoints (std::vector< int > *index_esqueleto,* unsigned short ∗ *vol_esqueleto,* std::vector< int > & *ptos_extremos,* int *max1,* int *max2,* int *max3*)

6.11.2.11 void LabelBranchs (unsigned short ∗ *vol_esqueleto,* int *max1,* int *max2,* int *max3*)

6.11.2.12 void LabelBranchsNew (unsigned short ∗ *vol_esqueleto,* int *max1,* int *max2,* int *max3*)

6.11.2.13 int main (int *argc,* char ∗ *argv[ ]*)

6.11.2.14 int mapIndex3D (int *r,* int *c,* int *z,* int *nr,* int *nc,* int *nz*)

6.11.2.15 int maptox (int *mapindex,* int *max1,* int *max2*)

6.11.2.16 int maptoy (int *mapindex,* int *max1,* int *max2*)

6.11.2.17 int maptoz (int *mapindex,* int *max1,* int *max2*)

6.11.2.18 void print_timing (FILE ∗ *fp,* struct timeval *start,* struct timeval *end*)

6.11.2.19 void Relabeling (unsigned short ∗ *output,* int *max1,* int *max2,* int *max3*)

6.11.2.20 void reserve_memory_triple_float (float ∗∗∗ *out,* int *max1,* int *max2,* int *max3*)

6.11.2.21 void reserve_memory_triple_int (int ∗∗∗ *out,* int *max1,* int *max2,* int *max3*)

6.11.3 Variable Documentation

6.11.3.1 int debug

6.11.3.2 int verbose

6.12 /home/costa/Data/Code/C/apps/imported/pgm2itkvol/itkvol2pgm.cxx File Reference

```
#include <stdio.h>
#include <stdint.h>
#include <sys/types.h>
#include <stdlib.h>
#include <assert.h>
#include <mcimage.h>
#include <mccodimage.h>
#include "itkImage.h"
#include "itkImageFileReader.h"
#include "itkImageRegionIterator.h"
```

Macros

- #define VERBOSE

Functions

- int main (int argc, char ∗∗argv)

6.12.1 Macro Definition Documentation

6.12.1.1 #define VERBOSE

6.12.2 Function Documentation

6.12.2.1 int main (int *argc,* char ∗∗ *argv*)

6.13 /home/costa/Data/Code/C/apps/imported/pgm2itkvol/mccodimage.h File Reference

This file hold the basic type declarations used in the C functions of Pink.

```
#include <stdint.h>
#include "mcimage.h"
```

Classes

- struct fcomplex

  Complex number, represented by floats.
- struct dcomplex

  Complex number, represented by doubles.
- struct xvimage

  The image class for the C functions.

Macros

- #define NDG_MAX 255 /∗ niveau de gris max ∗/
- #define NDG_MIN 0 /∗ niveau de gris min ∗/
- #define VFF_TYP_BIT 0 /∗ pixels are on or off (binary image)∗/
- #define VFF_TYP_1_BYTE 1 /∗ pixels are byte (uint8_t) ∗/
- #define VFF_TYP_2_BYTE 2 /∗ pixels are two byte (int16_t) ∗/
- #define VFF_TYP_4_BYTE 4 /∗ pixels are four byte (int32_t) ∗/
- #define VFF_TYP_FLOAT 5 /∗ pixels are float (single precision)∗/
- #define VFF_TYP_DOUBLE 6 /∗ pixels are float (double precision)∗/
- #define VFF_TYP_COMPLEX 7 /∗ pixels are complex (single precision)∗/
- #define VFF_TYP_DCOMPLEX 8 /∗ pixels are complex (double precision)∗/
- #define SCHARDATA(I) ((int8_t∗)((I)->image_data))
- #define UCHARDATA(I) ((uint8_t∗)((I)->image_data))
- #define SSHORTDATA(I) ((int16_t∗)((I)->image_data))
- #define USHORTDATA(I) ((uint16_t∗)((I)->image_data))
- #define SLONGDATA(I) ((int32_t∗)((I)->image_data))
- #define ULONGDATA(I) ((uint32_t∗)((I)->image_data))
- #define FLOATDATA(I) ((float∗)((I)->image_data))
- #define DOUBLEDATA(I) ((double∗)((I)->image_data))
- #define COMPLEXDATA(I) ((fcomplex∗)((I)->image_data))
- #define DCOMPLEXDATA(I) ((dcomplex∗)((I)->image_data))
- #define colsize(I) ((I)->col_size)
- #define rowsize(I) ((I)->row_size)

- #define depth(I) ((I)->depth_size)
- #define tsize(I) ((I)->time_size)
- #define nbands(I) ((I)->num_data_bands)
- #define datatype(I) ((I)->data_storage_type)
- #define pixel(I, x, y) (((uint8_t∗)((I)->image_data))[(y)∗(I)->row_size+(x)])
- #define voxel(I, x, y, z) (((uint8_t∗)((I)->image_data))[((z)∗(I)->col_size+(y))∗(I)->row_size+(x)])
- #define lpixel(I, x, y) (((uint32_t∗)((I)->image_data))[(y)∗(I)->row_size+(x)])
- #define lvoxel(I, x, y, z) (((uint32_t∗)((I)->image_data))[((z)∗(I)->col_size+(y))∗(I)->row_size+(x)])
- #define EST 0
- #define NORD 2
- #define OUEST 4
- #define SUD 6
- #define NORD_EST 1
- #define NORD_OUEST 3
- #define SUD_OUEST 5
- #define SUD_EST 7
- #define DEVANT 8
- #define DERRIERE 10
- #define nonbord(p, rs, N) ((p%rs!=rs-1)&&(p>=rs)&&(p%rs!=0)&&(p<N-rs))
- #define nonbord3d(p, rs, ps, N) ((p>=ps)&&(p<N-ps)&&(p%ps>=rs)&&(p%ps<ps-rs)&&(p%rs!=0)&&(p%rs!=rs-1))
- #define ACCEPTED_TYPES1(I, T0)
- #define ACCEPTED_TYPES2(I, T0, T1)
- #define ACCEPTED_TYPES3(I, T0, T1, T2)
- #define ACCEPTED_TYPES4(I, T0, T1, T2, T3)
- #define ACCEPTED_TYPES5(I, T0, T1, T2, T3, T4)
- #define ACCEPTED_TYPES6(I, T0, T1, T2, T3, T4, T5)
- #define ACCEPTED_TYPES7(I, T0, T1, T2, T3, T4, T5, T6)
- #define COMPARE_SIZE(I0, I1)
- #define ONLY_2D(I)
- #define ONLY_3D(I)

Typedefs

- typedef struct xvimage xvimage

Functions

- int32_t voisin (index_t i, int32_t k, index_t rs, index_t nb)
- int32_t voisin2 (index_t i, int32_t k, index_t rs, index_t nb)
- int32_t voisin6 (index_t i, int32_t k, index_t rs, index_t n, index_t nb)
- int32_t bord (index_t i, index_t rs, index_t nb)
- int32_t bord3d (index_t i, index_t rs, index_t ps, index_t nb)
- int32_t voisin26 (index_t i, int32_t k, index_t rs, index_t n, index_t nb)
- int32_t voisin18 (index_t i, int32_t k, index_t rs, index_t n, index_t nb)
- int32_t voisins4 (index_t i, index_t j, index_t rs)
- int32_t voisins8 (index_t i, index_t j, index_t rs)
- int32_t voisins6 (index_t i, index_t j, index_t rs, index_t ps)
- int32_t voisins18 (index_t i, index_t j, index_t rs, index_t ps)
- int32_t voisins26 (index_t i, index_t j, index_t rs, index_t ps)
- int32_t voisin5 (index_t i, int32_t k, index_t rs, index_t nb)
- int32_t voisin6b (index_t i, int32_t k, index_t rs, index_t nb, index_t par)
- int32_t voisinNESO (index_t i, int32_t k, index_t rs, index_t nb)
- int32_t voisinNOSE (index_t i, int32_t k, index_t rs, index_t nb)

- int32_t voisin14b (index_t i, int32_t k, index_t rs, index_t ps, index_t N)
- int32_t voisinONAV (index_t i, int32_t k, index_t rs, index_t ps, index_t N)
- int32_t voisinENAR (index_t i, int32_t k, index_t rs, index_t ps, index_t N)
- int32_t voisinENAV (index_t i, int32_t k, index_t rs, index_t ps, index_t N)
- int32_t voisinONAR (index_t i, int32_t k, index_t rs, index_t ps, index_t N)
- uint32_t maskvois26 (uint8_t ∗F, uint32_t bitmask, index_t i, index_t rs, index_t ps, index_t N)
- int32_t sont4voisins (index_t p, index_t q, index_t rs)
- int32_t sont8voisins (index_t p, index_t q, index_t rs)
- int32_t sont6voisins (index_t p, index_t q, index_t rs, index_t ps)
- int32_t sont18voisins (index_t p, index_t q, index_t rs, index_t ps)
- int32_t sont26voisins (index_t p, index_t q, index_t rs, index_t ps)
- int32_t voisin125 (index_t i, int32_t k, index_t rs, index_t ps, index_t N)

6.13.1 Detailed Description

This file hold the basic type declarations used in the C functions of Pink. Pink

Author

Michel Couprie

6.13.2 Macro Definition Documentation

6.13.2.1 #define ACCEPTED_TYPES1(*I*, *T0*)

Value:

```
if (datatype(I)!=T0)                                   \
  {                                                      \
  fprintf(stderr, "%s: bad image type\n", F_NAME);       \
  return 0;                                              \
  }
```

6.13.2.2 #define ACCEPTED_TYPES2(*I*, *T0*, *T1*)

Value:

```
if ( (datatype(I)!=T0) && (datatype(I)!=T1) )          \
  {                                                      \
  fprintf(stderr, "%s: bad image type\n", F_NAME);       \
  return 0;                                              \
  }
```

6.13.2.3 #define ACCEPTED_TYPES3(*I*, *T0*, *T1*, *T2*)

Value:

```
if ( (datatype(I)!=T0) && (datatype(I)!=T1) && (datatype
      (I)!=T2) )    \
  {                                                                       \
  fprintf(stderr, "%s: bad image type\n", F_NAME);                        \
  return 0;                                                               \
  }
```

6.13.2.4 #define ACCEPTED_TYPES4(*I*, *T0*, *T1*, *T2*, *T3*)

Value:

```
if ( (datatype(I)!=T0) && (datatype(I)!=T1) && (datatype
      (I)!=T2)      \
       && (datatype(I)!=T3) )
       \
  {                                                                     \
  fprintf(stderr, "%s: bad image type\n", F_NAME);                      \
  return 0;                                                             \
  }
```

6.13.2.5 #define ACCEPTED_TYPES5(*I*, *T0*, *T1*, *T2*, *T3*, *T4*)

Value:

```
if ( (datatype(I)!=T0) && (datatype(I)!=T1) && (datatype
      (I)!=T2)      \
       && (datatype(I)!=T3) && (datatype(I)!=T4) )
               \
  {                                                                     \
  fprintf(stderr, "%s: bad image type\n", F_NAME);                      \
  return 0;                                                             \
  }
```

6.13.2.6 #define ACCEPTED_TYPES6(*I*, *T0*, *T1*, *T2*, *T3*, *T4*, *T5*)

Value:

```
if ( (datatype(I)!=T0) && (datatype(I)!=T1) && (datatype
      (I)!=T2)      \
       && (datatype(I)!=T3) && (datatype(I)!=T4)  && (datatype
      (I)!=T5)) \
  {                                                                     \
  fprintf(stderr, "%s: bad image type\n", F_NAME);                      \
  return 0;                                                             \
  }
```

6.13.2.7 #define ACCEPTED_TYPES7(*I*, *T0*, *T1*, *T2*, *T3*, *T4*, *T5*, *T6*)

Value:

```
if ( (datatype(I)!=T0) && (datatype(I)!=T1) && (datatype
      (I)!=T2)      \
       && (datatype(I)!=T3) && (datatype(I)!=T4)  && (datatype
      (I)!=T5)  \
       && (datatype(I)!=T6) )
       \
  {                                                                     \
  fprintf(stderr, "%s: bad image type\n", F_NAME);                      \
  return 0;                                                             \
  }
```

6.13.2.8 #define colsize(*I*) ((I)->col_size)

6.13.2.9 #define COMPARE_SIZE(*I0*, *I1*)

Value:

```
if (rowsize(I0)!=rowsize(I1) && rowsize(I0)!=rowsize
      (I1) && depth(I0)!=depth(I1) \
      && (tsize(I0) != tsize(I1) ) && (nbands(I0) != nbands
      (I1) ) )                                  \
  {                                                                     \
  fprintf(stderr, "%s: incompatible image sizes\n", F_NAME);            \
  return 0;                                                             \
  }
```

6.13.2.10 #define COMPLEXDATA(*I*) ((fcomplex∗)((I)->image_data))

6.13.2.11 #define datatype(*I*) ((I)->data_storage_type)

6.13.2.12 #define DCOMPLEXDATA(*I*) ((dcomplex∗)((I)->image_data))

6.13.2.13 #define depth(*I*) ((I)->depth_size)

6.13.2.14 #define DERRIERE 10

6.13.2.15 #define DEVANT 8

6.13.2.16 #define DOUBLEDATA(*I*) ((double∗)((I)->image_data))

6.13.2.17 #define EST 0

6.13.2.18 #define FLOATDATA(*I*) ((float∗)((I)->image_data))

6.13.2.19 #define Ipixel(*I, x, y*) (((uint32_t∗)((I)->image_data))[(y)∗(I)->row_size+(x)])

6.13.2.20 #define Ivoxel(*I, x, y, z*) (((uint32_t∗)((I)->image_data))[((z)∗(I)->col_size+(y))∗(I)->row_size+(x)])

6.13.2.21 #define nbands(*I*) ((I)->num_data_bands)

6.13.2.22 #define NDG_MAX 255 /∗ niveau de gris max ∗/

6.13.2.23 #define NDG_MIN 0 /∗ niveau de gris min ∗/

6.13.2.24 #define nonbord(*p, rs, N*) ((p%rs!=rs-1)&&(p>=rs)&&(p%rs!=0)&&(p<N-rs))

6.13.2.25 #define nonbord3d(*p, rs, ps, N*) ((p>=ps)&&(p<N-ps)&&(p%ps>=rs)&&(p%ps<ps-rs)&&(p%rs!=0)&&(p%rs!=rs-1))

6.13.2.26 #define NORD 2

6.13.2.27 #define NORD_EST 1

6.13.2.28 #define NORD_OUEST 3

6.13.2.29 #define ONLY_2D(*I*)

Value:

```
if (depth(I)!=1)                                                \
  {                                                              \
  fprintf(stderr, "%s: only for 2D images\n", F_NAME);           \
  return 0;                                                      \
  }
```

6.13.2.30 #define ONLY_3D(*I*)

Value:

```
if (depth(I)==1)                                                \
  {                                                              \
  fprintf(stderr, "%s: only for 3D images\n", F_NAME);           \
  return 0;                                                      \
  }
```

6.13.2.31 #define OUEST 4

6.13.2.32 #define pixel(*I*, *x*, *y*) (((uint8_t∗)((I)->image_data))[(y)∗(I)->row_size+(x)])

6.13.2.33 #define rowsize(*I*) ((I)->row_size)

6.13.2.34 #define SCHARDATA(*I*) ((int8_t∗)((I)->image_data))

6.13.2.35 #define SLONGDATA(*I*) ((int32_t∗)((I)->image_data))

6.13.2.36 #define SSHORTDATA(*I*) ((int16_t∗)((I)->image_data))

6.13.2.37 #define SUD 6

6.13.2.38 #define SUD_EST 7

6.13.2.39 #define SUD_OUEST 5

6.13.2.40 #define tsize(*I*) ((I)->time_size)

6.13.2.41 #define UCHARDATA(*I*) ((uint8_t∗)((I)->image_data))

6.13.2.42 #define ULONGDATA(*I*) ((uint32_t∗)((I)->image_data))

6.13.2.43 #define USHORTDATA(*I*) ((uint16_t∗)((I)->image_data))

6.13.2.44 #define VFF_TYP_1_BYTE 1 /∗ pixels are byte (uint8_t) ∗/

6.13.2.45 #define VFF_TYP_2_BYTE 2 /∗ pixels are two byte (int16_t) ∗/

6.13.2.46 #define VFF_TYP_4_BYTE 4 /∗ pixels are four byte (int32_t) ∗/

6.13.2.47 #define VFF_TYP_BIT 0 /∗ pixels are on or off (binary image)∗/

6.13.2.48 #define VFF_TYP_COMPLEX 7 /∗ pixels are complex (single precision)∗/

6.13.2.49 #define VFF_TYP_DCOMPLEX 8 /∗ pixels are complex (double precision)∗/

6.13.2.50 #define VFF_TYP_DOUBLE 6 /∗ pixels are float (double precision)∗/

6.13.2.51 #define VFF_TYP_FLOAT 5 /∗ pixels are float (single precision)∗/

6.13.2.52 #define voxel(*I*, *x*, *y*, *z*) (((uint8_t∗)((I)->image_data))[((z)∗(I)->col_size+(y))∗(I)->row_size+(x)])

6.13.3 Typedef Documentation

6.13.3.1 typedef struct xvimage xvimage

6.13.4 Function Documentation

6.13.4.1 int32_t bord (index_t *i*, index_t *rs*, index_t *nb*)

6.13.4.2 int32_t bord3d (index_t *i*, index_t *rs*, index_t *ps*, index_t *nb*)

6.13.4.3 uint32_t maskvois26 (uint8_t ∗ *F*, uint32_t *bitmask*, index_t *i*, index_t *rs*, index_t *ps*, index_t *N*)

6.13.4.4 int32_t sont18voisins (index_t *p*, index_t *q*, index_t *rs*, index_t *ps*)

6.13.4.5 int32_t sont26voisins (index_t *p*, index_t *q*, index_t *rs*, index_t *ps*)

6.13.4.6 int32_t sont4voisins (index_t *p*, index_t *q*, index_t *rs*)

6.13.4.7 int32_t sont6voisins (index_t *p*, index_t *q*, index_t *rs*, index_t *ps*)

6.13.4.8 int32_t sont8voisins (index_t *p*, index_t *q*, index_t *rs*)

6.13.4.9 int32_t voisin (index_t *i*, int32_t *k*, index_t *rs*, index_t *nb*)

6.13.4.10 int32_t voisin125 (index_t *i*, int32_t *k*, index_t *rs*, index_t *ps*, index_t *N*)

6.13.4.11 int32_t voisin14b (index_t *i*, int32_t *k*, index_t *rs*, index_t *ps*, index_t *N*)

6.13.4.12 int32_t voisin18 (index_t *i*, int32_t *k*, index_t *rs*, index_t *n*, index_t *nb*)

6.13.4.13 int32_t voisin2 (index_t *i*, int32_t *k*, index_t *rs*, index_t *nb*)

6.13.4.14 int32_t voisin26 (index_t *i*, int32_t *k*, index_t *rs*, index_t *n*, index_t *nb*)

6.13.4.15 int32_t voisin5 (index_t *i*, int32_t *k*, index_t *rs*, index_t *nb*)

6.13.4.16 int32_t voisin6 (index_t *i*, int32_t *k*, index_t *rs*, index_t *n*, index_t *nb*)

6.13.4.17 int32_t voisin6b (index_t *i*, int32_t *k*, index_t *rs*, index_t *nb*, index_t *par*)

6.13.4.18 int32_t voisinENAR (index_t *i*, int32_t *k*, index_t *rs*, index_t *ps*, index_t *N*)

6.13.4.19 int32_t voisinENAV (index_t *i*, int32_t *k*, index_t *rs*, index_t *ps*, index_t *N*)

6.13.4.20 int32_t voisinNESO (index_t *i*, int32_t *k*, index_t *rs*, index_t *nb*)

6.13.4.21 int32_t voisinNOSE (index_t *i*, int32_t *k*, index_t *rs*, index_t *nb*)

6.13.4.22 int32_t voisinONAR (index_t *i*, int32_t *k*, index_t *rs*, index_t *ps*, index_t *N*)

6.13.4.23 int32_t voisinONAV (index_t *i*, int32_t *k*, index_t *rs*, index_t *ps*, index_t *N*)

6.13.4.24 int32_t voisins18 (index_t *i*, index_t *j*, index_t *rs*, index_t *ps*)

6.13.4.25 int32_t voisins26 (index_t *i*, index_t *j*, index_t *rs*, index_t *ps*)

6.13.4.26 int32_t voisins4 (index_t *i*, index_t *j*, index_t *rs*)

6.13.4.27 int32_t voisins6 (index_t *i*, index_t *j*, index_t *rs*, index_t *ps*)

6.13.4.28 int32_t voisins8 (index_t *i*, index_t *j*, index_t *rs*)

6.14 /home/costa/Data/Code/C/apps/imported/pgm2itkvol/mcimage.c File Reference

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <string.h>
#include <ctype.h>
#include <assert.h>
#include <mcutil.h>
#include <mcimage.h>
#include <mccodimage.h>
```

Classes

- struct BITMAPFILEHEADER
- struct BITMAPINFOHEADER
- struct RGBFILEHEADER

Macros

- #define INT32_MAX (2147483647)
- #define BUFFERSIZE 10000
- #define WARN_HUGE
- #define F_NAME "allocimage"
- #define F_NAME "allocmultimage"
- #define F_NAME "razimage"
- #define F_NAME "allocheader"
- #define F_NAME "showheader"
- #define F_NAME "copyimage"
- #define F_NAME "copy2image"
- #define F_NAME "equalimages"
- #define F_NAME "convertgen"
- #define F_NAME "convertlong"
- #define F_NAME "convertfloat"
- #define F_NAME "list2image"
- #define F_NAME "image2list"
- #define F_NAME "writeimage"
- #define F_NAME "writerawimage"
- #define F_NAME "writese"
- #define F_NAME "writeascimage"
- #define F_NAME "printimage"
- #define F_NAME "writergbimage"
- #define F_NAME "writergbascimage"
- #define F_NAME "writelongimage"
- #define F_NAME "readimage"
- #define F_NAME "readheader"
- #define F_NAME "readse"
- #define F_NAME "readrgbimage"
- #define F_NAME "readlongimage"
- #define F_NAME "readbmp"
- #define F_NAME "writebmp"

- #define F_NAME "readrgb"
- #define F_NAME "writelist2"
- #define F_NAME "writelist3"

Functions

- __pink__inline FILE ∗ pink_fopen_read (char ∗filename)
- __pink__inline FILE ∗ pink_fopen_write (char ∗filename)
- struct xvimage ∗ allocimage (char ∗name, index_t rs, index_t cs, index_t ds, int32_t dt)

  Allocates an image object with the given size and type.
- struct xvimage ∗ allocmultimage (char ∗name, index_t rs, index_t cs, index_t ds, index_t ts, index_t nb, int32_t dt)
- void razimage (struct xvimage ∗f)

  fills the image with zeros description Sets every pixel of the image to binary zero.
- struct xvimage ∗ allocheader (char ∗name, index_t rs, index_t cs, index_t d, int32_t t)
- int32_t showheader (char ∗name)
- void freeimage (struct xvimage ∗image)

  Frees an image object.
- struct xvimage ∗ copyimage (struct xvimage ∗f)
- int32_t copy2image (struct xvimage ∗dest, struct xvimage ∗source)
- int32_t equalimages (struct xvimage ∗im1, struct xvimage ∗im2)
- int32_t convertgen (struct xvimage ∗∗f1, struct xvimage ∗∗f2)
- int32_t convertlong (struct xvimage ∗∗f1)
- int32_t convertfloat (struct xvimage ∗∗f1)
- void list2image (struct xvimage ∗image, double ∗P, index_t n)
- double ∗ image2list (struct xvimage ∗image, index_t ∗n)
- void writeimage (struct xvimage ∗image, char ∗filename)

  Writes an image to disk.
- void writerawimage (struct xvimage ∗image, char ∗filename)
- void writese (struct xvimage ∗image, char ∗filename, index_t x, index_t y, index_t z)
- void writeascimage (struct xvimage ∗image, char ∗filename)
- void printimage (struct xvimage ∗image)
- void writergbimage (struct xvimage ∗redimage, struct xvimage ∗greenimage, struct xvimage ∗blueimage, char ∗filename)
- void writergbascimage (struct xvimage ∗redimage, struct xvimage ∗greenimage, struct xvimage ∗blueimage, char ∗filename)
- void writelongimage (struct xvimage ∗image, char ∗filename)
- struct xvimage ∗ readimage (const char ∗filename)

  Reads an image from a file.
- struct xvimage ∗ readheader (char ∗filename)
- struct xvimage ∗ readse (char ∗filename, index_t ∗x, index_t ∗y, index_t ∗z)
- int32_t readrgbimage (char ∗filename, struct xvimage ∗∗r, struct xvimage ∗∗g, struct xvimage ∗∗b)
- struct xvimage ∗ readlongimage (char ∗filename)
- void freadushort (uint16_t ∗ptr, FILE ∗fd)
- void freadulong (uint32_t ∗ptr, FILE ∗fd)
- int32_t readbmp (char ∗filename, struct xvimage ∗∗r, struct xvimage ∗∗g, struct xvimage ∗∗b)
- void fwriteushort (uint16_t us, FILE ∗fd)
- void fwriteulong (uint32_t ul, FILE ∗fd)
- void writebmp (struct xvimage ∗redimage, struct xvimage ∗greenimage, struct xvimage ∗blueimage, char ∗filename)
- int32_t readrgb (char ∗filename, struct xvimage ∗∗r, struct xvimage ∗∗g, struct xvimage ∗∗b)
- void writelist2 (char ∗filename, int32_t ∗x, int32_t ∗y, int32_t npoints)
- void writelist3 (char ∗filename, int32_t ∗x, int32_t ∗y, int32_t ∗z, int32_t npoints)

6.14.1 Macro Definition Documentation

6.14.1.1 #define BUFFERSIZE 10000

6.14.1.2 #define F_NAME "allocimage"

6.14.1.3 #define F_NAME "allocmultimage"

6.14.1.4 #define F_NAME "razimage"

6.14.1.5 #define F_NAME "allocheader"

6.14.1.6 #define F_NAME "showheader"

6.14.1.7 #define F_NAME "copyimage"

6.14.1.8 #define F_NAME "copy2image"

6.14.1.9 #define F_NAME "equalimages"

6.14.1.10 #define F_NAME "convertgen"

6.14.1.11 #define F_NAME "convertlong"

6.14.1.12 #define F_NAME "convertfloat"

6.14.1.13 #define F_NAME "list2image"

6.14.1.14 #define F_NAME "image2list"

6.14.1.15 #define F_NAME "writeimage"

6.14.1.16 #define F_NAME "writerawimage"

6.14.1.17 #define F_NAME "writese"

6.14.1.18 #define F_NAME "writeascimage"

6.14.1.19 #define F_NAME "printimage"

6.14.1.20 #define F_NAME "writergbimage"

6.14.1.21 #define F_NAME "writergbascimage"

6.14.1.22 #define F_NAME "writelongimage"

6.14.1.23 #define F_NAME "readimage"

6.14.1.24 #define F_NAME "readheader"

6.14.1.25 #define F_NAME "readse"

6.14.1.26 #define F_NAME "readrgbimage"

6.14.1.27 #define F_NAME "readlongimage"

6.14.1.28 #define F_NAME "readbmp"

6.14.1.29 #define F_NAME "writebmp"

6.14.1.30 #define F_NAME "readrgb"

6.14.1.31 #define F_NAME "writelist2"

6.14.1.32 #define F_NAME "writelist3"

6.14.1.33 #define INT32_MAX (2147483647)

6.14.1.34 #define WARN_HUGE

6.14.2 Function Documentation

6.14.2.1 struct xvimage∗ allocheader (char ∗ *name*, index_t *rs*, index_t *cs*, index_t *d*, int32_t *t*) [read]

6.14.2.2 struct xvimage∗ allocimage (char ∗ *name*, index_t *rs*, index_t *cs*, index_t *ds*, int32_t *t*) [read]

Allocates an image object with the given size and type.

Parameters

name	Not used
rs	x-size
cs	y-size
ds	z-size
t	t-size

Returns

The pointer to the image.

6.14.2.3 struct xvimage∗ allocmultimage (char ∗ *name*, index_t *rs*, index_t *cs*, index_t *ds*, index_t *ts*, index_t *nb*, int32_t *dt*) [read]

6.14.2.4 int32_t convertfloat (struct xvimage ∗∗ *f1*)

6.14.2.5 int32_t convertgen (struct xvimage ∗∗ *f1*, struct xvimage ∗∗ *f2*)

6.14.2.6 int32_t convertlong (struct xvimage ∗∗ *f1*)

6.14.2.7 int32_t copy2image (struct xvimage ∗ *dest*, struct xvimage ∗ *source*)

6.14.2.8 struct xvimage∗ copyimage (struct xvimage ∗ *f*) [read]

6.14.2.9 int32_t equalimages (struct xvimage ∗ *im1*, struct xvimage ∗ *im2*)

6.14.2.10 void freadulong (uint32_t ∗ *ptr*, FILE ∗ *fd*)

6.14.2.11 void freadushort (uint16_t ∗ *ptr*, FILE ∗ *fd*)

6.14.2.12 void freeimage (struct xvimage ∗ *image*)

Frees an image object.

Parameters

image	The pointer to the image

6.14.2.13 void fwriteulong (uint32_t *ul*, FILE ∗ *fd*)

6.14.2.14 void fwriteushort (uint16_t *us*, FILE ∗ *fd*)

6.14.2.15 double∗ image2list (struct xvimage ∗ *image*, index_t ∗ *n*)

6.14.2.16 void list2image (struct xvimage ∗ *image*, double ∗ *P*, index_t *n*)

6.14.2.17 __pink__inline FILE∗ pink_fopen_read (char ∗ *filename*)

6.14.2.18 __pink__inline FILE∗ pink_fopen_write (char ∗ *filename*)

6.14.2.19 void printimage (struct xvimage ∗ *image*)

6.14.2.20 void razimage (struct xvimage ∗ *f*)

fills the image with zeros description Sets every pixel of the image to binary zero.

Parameters

f	the input image

Returns

no return value

6.14.2.21 int32_t readbmp (char ∗ *filename*, struct xvimage ∗∗ *r*, struct xvimage ∗∗ *g*, struct xvimage ∗∗ *b*)

6.14.2.22 struct xvimage∗ readheader (char ∗ *filename*) [read]

6.14.2.23 struct xvimage∗ readimage (const char ∗ *filename*) [read]

Reads an image from a file.

Parameters

filename	The name of the image file.

Returns

A Pointer to a newly allocated image.

6.14.2.24 struct xvimage∗ readlongimage (char ∗ *filename*) [read]

6.14.2.25 int32_t readrgb (char ∗ *filename*, struct xvimage ∗∗ *r*, struct xvimage ∗∗ *g*, struct xvimage ∗∗ *b*)

plus 404 bytes dummy padding to make header 512 bytes padding bytes

red bytes

green bytes

blue bytes

6.14.2.26 int32_t readrgbimage (char ∗ *filename,* struct xvimage ∗∗ *r,* struct xvimage ∗∗ *g,* struct xvimage ∗∗ *b*)

6.14.2.27 struct xvimage∗ readse (char ∗ *filename,* index_t ∗ *x,* index_t ∗ *y,* index_t ∗ *z*) [read]

6.14.2.28 int32_t showheader (char ∗ *name*)

6.14.2.29 void writeascimage (struct xvimage ∗ *image,* char ∗ *filename*)

6.14.2.30 void writebmp (struct xvimage ∗ *redimage,* struct xvimage ∗ *greenimage,* struct xvimage ∗ *blueimage,* char ∗ *filename*)

6.14.2.31 void writeimage (struct xvimage ∗ *image,* char ∗ *filename*)

Writes an image to disk.

Parameters

image	The pointer to the image
filename	The file to write the image at.

6.14.2.32 void writelist2 (char ∗ *filename,* int32_t ∗ *x,* int32_t ∗ *y,* int32_t *npoints*)

6.14.2.33 void writelist3 (char ∗ *filename,* int32_t ∗ *x,* int32_t ∗ *y,* int32_t ∗ *z,* int32_t *npoints*)

6.14.2.34 void writelongimage (struct xvimage ∗ *image,* char ∗ *filename*)

6.14.2.35 void writerawimage (struct xvimage ∗ *image,* char ∗ *filename*)

6.14.2.36 void writergbascimage (struct xvimage ∗ *redimage,* struct xvimage ∗ *greenimage,* struct xvimage ∗ *blueimage,* char ∗ *filename*)

6.14.2.37 void writergbimage (struct xvimage ∗ *redimage,* struct xvimage ∗ *greenimage,* struct xvimage ∗ *blueimage,* char ∗ *filename*)

6.14.2.38 void writese (struct xvimage ∗ *image,* char ∗ *filename,* index_t *x,* index_t *y,* index_t *z*)

6.15 /home/costa/Data/Code/C/apps/imported/pgm2itkvol/mcimage.h File Reference

This file holds the basic image allocation functions.

Macros

- #define __pink__inline
- #define HUGE_IMAGE_SIZE INT32_MAX

Typedefs

- typedef unsigned char u_int8_t
- typedef unsigned int u_int32_t
- typedef unsigned char uint8_t
- typedef int int32_t
- typedef unsigned int uint32_t
- typedef int32_t index_t

Functions

- struct xvimage ∗ allocimage (char ∗name, index_t rs, index_t cs, index_t ds, int32_t t)

  Allocates an image object with the given size and type.
- struct xvimage ∗ allocmultimage (char ∗name, index_t rs, index_t cs, index_t ds, index_t ts, index_t nb, int32_t t)
- void razimage (struct xvimage ∗f)

  fills the image with zeros description Sets every pixel of the image to binary zero.
- struct xvimage ∗ allocheader (char ∗name, index_t rs, index_t cs, index_t d, int32_t t)
- int32_t showheader (char ∗name)
- void freeimage (struct xvimage ∗image)

  Frees an image object.
- struct xvimage ∗ copyimage (struct xvimage ∗f)
- int32_t copy2image (struct xvimage ∗dest, struct xvimage ∗source)
- int32_t equalimages (struct xvimage ∗im1, struct xvimage ∗im2)
- void list2image (struct xvimage ∗image, double ∗P, index_t n)
- double ∗ image2list (struct xvimage ∗image, index_t ∗n)
- void writeimage (struct xvimage ∗image, char ∗filename)

  Writes an image to disk.
- void writese (struct xvimage ∗image, char ∗filename, index_t x, index_t y, index_t z)
- void writelongimage (struct xvimage ∗image, char ∗filename)
- void writerawimage (struct xvimage ∗image, char ∗filename)
- void writeascimage (struct xvimage ∗image, char ∗filename)
- void printimage (struct xvimage ∗image)
- void writergbimage (struct xvimage ∗redimage, struct xvimage ∗greenimage, struct xvimage ∗blueimage, char ∗filename)
- void writergbascimage (struct xvimage ∗redimage, struct xvimage ∗greenimage, struct xvimage ∗blueimage, char ∗filename)
- struct xvimage ∗ readimage (const char ∗filename)

  Reads an image from a file.
- struct xvimage ∗ readheader (char ∗filename)
- struct xvimage ∗ readse (char ∗filename, index_t ∗x, index_t ∗y, index_t ∗z)
- struct xvimage ∗ readlongimage (char ∗filename)
- int32_t readrgbimage (char ∗filename, struct xvimage ∗∗r, struct xvimage ∗∗g, struct xvimage ∗∗b)
- int32_t readbmp (char ∗filename, struct xvimage ∗∗r, struct xvimage ∗∗g, struct xvimage ∗∗b)
- void writebmp (struct xvimage ∗redimage, struct xvimage ∗greenimage, struct xvimage ∗blueimage, char ∗filename)
- int32_t readrgb (char ∗filename, struct xvimage ∗∗r, struct xvimage ∗∗g, struct xvimage ∗∗b)
- int32_t convertgen (struct xvimage ∗∗f1, struct xvimage ∗∗f2)
- int32_t convertlong (struct xvimage ∗∗f1)
- int32_t convertfloat (struct xvimage ∗∗f1)
- void writelist2 (char ∗filename, int32_t ∗x, int32_t ∗y, int32_t npoints)
- void writelist3 (char ∗filename, int32_t ∗x, int32_t ∗y, int32_t ∗z, int32_t npoints)

6.15.1 Detailed Description

This file holds the basic image allocation functions. Pink

Author

Michel Couprie, 2009

6.15.2 Macro Definition Documentation

6.15.2.1 #define __pink__inline

6.15.2.2 #define HUGE_IMAGE_SIZE INT32_MAX

6.15.3 Typedef Documentation

6.15.3.1 typedef int32_t index_t

6.15.3.2 typedef int int32_t

6.15.3.3 typedef unsigned int u_int32_t

6.15.3.4 typedef unsigned char u_int8_t

6.15.3.5 typedef unsigned int uint32_t

6.15.3.6 typedef unsigned char uint8_t

6.15.4 Function Documentation

6.15.4.1 struct xvimage∗ allocheader (char ∗ *name*, index_t *rs*, index_t *cs*, index_t *d*, int32_t *t*) `[read]`

6.15.4.2 struct xvimage∗ allocimage (char ∗ *name*, index_t *rs*, index_t *cs*, index_t *ds*, int32_t *t*) `[read]`

Allocates an image object with the given size and type.

Parameters

name	Not used
rs	x-size
cs	y-size
ds	z-size
t	t-size

Returns

The pointer to the image.

6.15.4.3 struct xvimage∗ allocmultimage (char ∗ *name*, index_t *rs*, index_t *cs*, index_t *ds*, index_t *ts*, index_t *nb*, int32_t *t*) `[read]`

6.15.4.4 int32_t convertfloat (struct xvimage ∗∗ *f1*)

6.15.4.5 int32_t convertgen (struct xvimage ∗∗ *f1*, struct xvimage ∗∗ *f2*)

6.15.4.6 int32_t convertlong (struct xvimage ∗∗ *f1*)

6.15.4.7 int32_t copy2image (struct xvimage ∗ *dest*, struct xvimage ∗ *source*)

6.15.4.8 struct xvimage∗ copyimage (struct xvimage ∗ *f*) `[read]`

6.15.4.9 int32_t equalimages (struct xvimage ∗ *im1*, struct xvimage ∗ *im2*)

6.15.4.10 void freeimage (struct xvimage * *image*)

Frees an image object.

Parameters

image	The pointer to the image

6.15.4.11 double* image2list (struct xvimage * *image,* index_t * *n*)

6.15.4.12 void list2image (struct xvimage * *image,* double * *P,* index_t *n*)

6.15.4.13 void printimage (struct xvimage * *image*)

6.15.4.14 void razimage (struct xvimage * *f*)

fills the image with zeros description Sets every pixel of the image to binary zero.

Parameters

f	the input image

Returns

no return value

6.15.4.15 int32_t readbmp (char * *filename,* struct xvimage ** *r,* struct xvimage ** *g,* struct xvimage ** *b*)

6.15.4.16 struct xvimage* readheader (char * *filename*) [read]

6.15.4.17 struct xvimage* readimage (const char * *filename*) [read]

Reads an image from a file.

Parameters

filename	The name of the image file.

Returns

A Pointer to a newly allocated image.

6.15.4.18 struct xvimage* readlongimage (char * *filename*) [read]

6.15.4.19 int32_t readrgb (char * *filename,* struct xvimage ** *r,* struct xvimage ** *g,* struct xvimage ** *b*)

plus 404 bytes dummy padding to make header 512 bytes padding bytes

red bytes

green bytes

blue bytes

6.15.4.20 int32_t readrgbimage (char * *filename,* struct xvimage ** *r,* struct xvimage ** *g,* struct xvimage ** *b*)

6.15.4.21 struct xvimage∗ readse (char ∗ *filename,* index_t ∗ *x,* index_t ∗ *y,* index_t ∗ *z*) [read]

6.15.4.22 int32_t showheader (char ∗ *name*)

6.15.4.23 void writeascimage (struct xvimage ∗ *image,* char ∗ *filename*)

6.15.4.24 void writebmp (struct xvimage ∗ *redimage,* struct xvimage ∗ *greenimage,* struct xvimage ∗ *blueimage,* char ∗ *filename*)

6.15.4.25 void writeimage (struct xvimage ∗ *image,* char ∗ *filename*)

Writes an image to disk.

Parameters

image	The pointer to the image
filename	The file to write the image at.

6.15.4.26 void writelist2 (char ∗ *filename,* int32_t ∗ *x,* int32_t ∗ *y,* int32_t *npoints*)

6.15.4.27 void writelist3 (char ∗ *filename,* int32_t ∗ *x,* int32_t ∗ *y,* int32_t ∗ *z,* int32_t *npoints*)

6.15.4.28 void writelongimage (struct xvimage ∗ *image,* char ∗ *filename*)

6.15.4.29 void writerawimage (struct xvimage ∗ *image,* char ∗ *filename*)

6.15.4.30 void writergbascimage (struct xvimage ∗ *redimage,* struct xvimage ∗ *greenimage,* struct xvimage ∗ *blueimage,* char ∗ *filename*)

6.15.4.31 void writergbimage (struct xvimage ∗ *redimage,* struct xvimage ∗ *greenimage,* struct xvimage ∗ *blueimage,* char ∗ *filename*)

6.15.4.32 void writese (struct xvimage ∗ *image,* char ∗ *filename,* index_t *x,* index_t *y,* index_t *z*)

6.16 /home/costa/Data/Code/C/apps/imported/pgm2itkvol/mcutil.h File Reference

Macros

- #define mcabs(X) ((X)>=0?(X):-(X))
- #define mcmax(X, Y) ((X)>=(Y)?(X):(Y))
- #define mcmin(X, Y) ((X)<=(Y)?(X):(Y))
- #define mcodd(X) ((X)&1)
- #define mceven(X) (((X)&1)==0)
- #define arrondi(z) (((z)-(double)((int32_t)(z)))<=0.5?((int32_t)(z)):((int32_t)(z+1)))
- #define signe(z) (((z)>0.0)?1.0:-1.0)
- #define mcsqr(x) ((x)∗(x))
- #define M_E 2.7182818284590452354 /∗ e ∗/
- #define M_LOG2E 1.4426950408889634074 /∗ log_2 e ∗/
- #define M_LOG10E 0.43429448190325182765 /∗ log_10 e ∗/
- #define M_LN2 0.69314718055994530942 /∗ log_e 2 ∗/
- #define M_LN10 2.30258509299404568402 /∗ log_e 10 ∗/
- #define M_PI 3.14159265358979323846 /∗ pi ∗/
- #define M_PI_2 1.57079632679489661923 /∗ pi/2 ∗/
- #define M_PI_4 0.78539816339744830962 /∗ pi/4 ∗/

- #define M_1_PI 0.31830988618379067154 /∗ 1/pi ∗/
- #define M_2_PI 0.63661977236758134308 /∗ 2/pi ∗/
- #define M_2_SQRTPI 1.12837916709551257390 /∗ 2/sqrt(pi) ∗/
- #define M_SQRT2 1.41421356237309504880 /∗ sqrt(2) ∗/
- #define M_SQRT1_2 0.70710678118654752440 /∗ 1/sqrt(2) ∗/

6.16.1 Macro Definition Documentation

6.16.1.1 #define arrondi(*z*) (((z)-(double)((int32_t)(z)))<=0.5?((int32_t)(z)):((int32_t)(z+1)))

6.16.1.2 #define M_1_PI 0.31830988618379067154 /∗ 1/pi ∗/

6.16.1.3 #define M_2_PI 0.63661977236758134308 /∗ 2/pi ∗/

6.16.1.4 #define M_2_SQRTPI 1.12837916709551257390 /∗ 2/sqrt(pi) ∗/

6.16.1.5 #define M_E 2.7182818284590452354 /∗ e ∗/

6.16.1.6 #define M_LN10 2.30258509299404568402 /∗ log_e 10 ∗/

6.16.1.7 #define M_LN2 0.69314718055994530942 /∗ log_e 2 ∗/

6.16.1.8 #define M_LOG10E 0.43429448190325182765 /∗ log_10 e ∗/

6.16.1.9 #define M_LOG2E 1.4426950408889634074 /∗ log_2 e ∗/

6.16.1.10 #define M_PI 3.14159265358979323846 /∗ pi ∗/

6.16.1.11 #define M_PI_2 1.57079632679489661923 /∗ pi/2 ∗/

6.16.1.12 #define M_PI_4 0.78539816339744830962 /∗ pi/4 ∗/

6.16.1.13 #define M_SQRT1_2 0.70710678118654752440 /∗ 1/sqrt(2) ∗/

6.16.1.14 #define M_SQRT2 1.41421356237309504880 /∗ sqrt(2) ∗/

6.16.1.15 #define mcabs(*X*) ((X)>=0?(X):-(X))

6.16.1.16 #define mceven(*X*) (((X)&1)==0)

6.16.1.17 #define mcmax(*X,* *Y*) ((X)>=(Y)?(X):(Y))

6.16.1.18 #define mcmin(*X,* *Y*) ((X)<=(Y)?(X):(Y))

6.16.1.19 #define mcodd(*X*) ((X)&1)

6.16.1.20 #define mcsqr(*x*) ((x)∗(x))

6.16.1.21 #define signe(*z*) (((z)>0.0)?1.0:-1.0)

6.17 /home/costa/Data/Code/C/apps/imported/pgm2itkvol/pgm2itkvol.cxx File Reference

```
#include <stdio.h>
#include <stdint.h>
#include <sys/types.h>
#include <stdlib.h>
#include <mcimage.h>
#include <mccodimage.h>
#include <itkImage.h>
#include <itkImageRegionIterator.h>
#include <itkImageFileWriter.h>
```

Macros

- #define VERBOSE
- #define UINTDATA unsigned int ∗

Functions

- int main (int argc, char ∗∗argv)

6.17.1 Macro Definition Documentation

6.17.1.1 #define UINTDATA unsigned int ∗

6.17.1.2 #define VERBOSE

6.17.2 Function Documentation

6.17.2.1 int main (int *argc,* char ∗∗ *argv*)

6.18 /home/costa/Data/Code/C/apps/LabelBiventricularMesh.cpp File Reference

Generate labels for Rafa's biventricular mesh.

```
#include <vtkPolyData.h>
#include <vtkDataSet.h>
#include <vtkAppendPolyData.h>
#include <vtkSmartPointer.h>
#include "vtkTransform.h"
#include "vtkTransformPolyDataFilter.h"
#include "vtkPolyDataNormals.h"
#include "vtkPolyDataConnectivityFilter.h"
#include "vtkShortArray.h"
#include "vtkCellData.h"
#include "vtkCellLocator.h"
#include "vtkCell.h"
#include "vtkPointData.h"
#include "CommonTools.h"
#include "vnl/vnl_vector.h"
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
```

Functions

- void SetScalars (vtkPolyData ∗mesh, int value, const char ∗array)

  add scalar array to the mesh. Don't remember why it is not in common tools
- int main (int argc, char ∗∗argv)

6.18.1 Detailed Description

Generate labels for Rafa's biventricular mesh.

6.18.2 Function Documentation

6.18.2.1 int main (int *argc,* char ∗∗ *argv*)

6.18.2.2 void SetScalars (vtkPolyData ∗ *mesh,* int *value,* const char ∗ *array*)

add scalar array to the mesh. Don't remember why it is not in common tools

6.19 /home/costa/Data/Code/C/apps/MakeBiventricularMesh.cpp File Reference

Make biventricular mesh for Rafa's model.

```
#include <vtkPolyData.h>
#include <vtkPolyDataToImageStencil.h>
#include <vtkImageStencil.h>
#include <vtkImageData.h>
#include <vtkDataSet.h>
#include <vtkDataSetReader.h>
#include <vtkDataSetWriter.h>
#include <vtkAppendPolyData.h>
#include <vtkSmartPointer.h>
#include "vtkTransform.h"
#include "vtkTransformPolyDataFilter.h"
#include "vtkPolyDataNormals.h"
#include "vtkImageWeightedSum.h"
#include "vtkImageOpenClose3D.h"
#include "vtkImageMarchingCubes.h"
#include "vtkImageContinuousErode3D.h"
#include "vtkImageGaussianSmooth.h"
#include "vtkSmoothPolyDataFilter.h"
#include "CommonTools.h"
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
```

Functions

- vtkImageData ∗ CreateEmptyImage (double ∗bounds, double ∗spacing, double padding, float value)

  creates an empty image
- vtkImageData ∗ CreateMask (vtkImageData ∗image, vtkPolyData ∗shape, float value)

  creates a mask
- vtkPolyData ∗ DecimateMesh (vtkPolyData ∗mesh, int nFaces)

decimates mesh

- int main (int argc, char ∗∗argv)

6.19.1 Detailed Description

Make biventricular mesh for Rafa's model.

6.19.2 Function Documentation

6.19.2.1 vtkImageData ∗ CreateEmptyImage (double ∗ *bounds,* double ∗ *spacing,* double *padding,* float *value*)

creates an empty image

6.19.2.2 vtkImageData ∗ CreateMask (vtkImageData ∗ *image,* vtkPolyData ∗ *shape,* float *value*)

creates a mask

6.19.2.3 vtkPolyData ∗ DecimateMesh (vtkPolyData ∗ *mesh,* int *nFaces*)

decimates mesh

6.19.2.4 int main (int *argc,* char ∗∗ *argv*)

6.20 /home/costa/Data/Code/C/apps/MeshHeart.cpp File Reference

Something to do with biventricular model generation.

```
#include <vtkPolyData.h>
#include <vtkPolyDataToImageStencil.h>
#include <vtkImageStencil.h>
#include <vtkImageData.h>
#include <vtkDataSet.h>
#include <vtkDataSetReader.h>
#include <vtkDataSetWriter.h>
#include <vtkCellArray.h>
#include <vtkSmartPointer.h>
#include "vtkTransform.h"
#include "vtkTransformPolyDataFilter.h"
#include "CommonTools.h"
#include <vtkDelaunay3D.h>
#include <vtkType.h>
#include <vtkUnstructuredGrid.h>
#include <vtkPointData.h>
#include <vtkShortArray.h>
#include <vtkThreshold.h>
#include <vtkDataSetSurfaceFilter.h>
#include <vtkCellData.h>
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
```

Functions

- vtkPolyData ∗ ExtractSurface (vtkUnstructuredGrid ∗volmesh, int id)
- int main (int argc, char ∗∗argv)

6.20.1 Detailed Description

Something to do with biventricular model generation.

6.20.2 Function Documentation

6.20.2.1 vtkPolyData ∗ ExtractSurface (vtkUnstructuredGrid ∗ *volmesh,* int *id*)

6.20.2.2 int main (int *argc,* char ∗∗ *argv*)

6.21 /home/costa/Data/Code/C/apps/MeshSegmentationLaplace.cpp File Reference

Mesh segmentation using distance to skeleton along the gradiwent of the solution of the laplacian equation.

```
#include "CommonTools.h"
#include <vtkSmartPointer.h>
#include <vtkPolyData.h>
#include <vtkPointData.h>
#include <vtkFloatArray.h>
#include <vtkType.h>
#include <vtkImageGradient.h>
#include <vtkAssignAttribute.h>
#include <vtkStreamTracer.h>
#include <vtkImageContinuousDilate3D.h>
#include <iostream>
```

Functions

- int main (int argc, char ∗argv[ ])

6.21.1 Detailed Description

Mesh segmentation using distance to skeleton along the gradiwent of the solution of the laplacian equation.

6.21.2 Function Documentation

6.21.2.1 int main (int *argc,* char ∗ *argv[ ]*)

6.22 /home/costa/Data/Code/C/apps/MRIRemesh.cpp File Reference

From shortaxis contours in MRI + reference point creates a smoother mesh using splines longitudinally.

```
#include <vtkPolyData.h>
#include <vtkCellArray.h>
#include <vtkSmartPointer.h>
#include "CommonTools.h"
#include <vtkPlane.h>
#include <vtkType.h>
#include <vtkPointData.h>
#include <vtkShortArray.h>
#include <vtkCellData.h>
#include <vtkCutter.h>
#include <vtkAppendPolyData.h>
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include <math.h>
#include <vector>
#include <vnl/vnl_vector.h>
#include <vnl/vnl_matrix.h>
#include <vnl/vnl_cross.h>
```

Classes

- struct sorting_struct

Macros

- #define CONTOUR_ENDO_APICAL 0
- #define CONTOUR_ENDO_BASAL 2
- #define CONTOUR_ENDO 1
- #define CONTOUR_EPI_APICAL 5
- #define CONTOUR_EPI_BASAL 7
- #define CONTOUR_EPI 6
- #define REF_POINT 10

Functions

- int struct_cmp_by_value (const void ∗a, const void ∗b)
- void ComputeCentroid (vtkPolyData ∗shape, double ∗centroid)

  compute centroid of polydata
- vnl_vector< double > ArbitraryRotate (vnl_vector< double > p, double theta, vnl_vector< double > p1, vnl_vector< double > p2)

  Rotate a point p by angle theta around an arbitrary line segment p1-p2.
- vtkPolyData ∗ OrderPoints (vtkPolyData ∗pts, int apical_id, int basal_id)

  Something about point order for this specific application.
- int main (int argc, char ∗∗argv)

6.22.1 Detailed Description

From shortaxis contours in MRI + reference point creates a smoother mesh using splines longitudinally.

6.22.2 Macro Definition Documentation

6.22.2.1 #define CONTOUR_ENDO 1

6.22.2.2 #define CONTOUR_ENDO_APICAL 0

6.22.2.3 #define CONTOUR_ENDO_BASAL 2

6.22.2.4 #define CONTOUR_EPI 6

6.22.2.5 #define CONTOUR_EPI_APICAL 5

6.22.2.6 #define CONTOUR_EPI_BASAL 7

6.22.2.7 #define REF_POINT 10

6.22.3 Function Documentation

6.22.3.1 vnl_vector< double > ArbitraryRotate (vnl_vector< double > *p,* double *theta,* vnl_vector< double > *p1,* vnl_vector< double > *p2*)

Rotate a point p by angle theta around an arbitrary line segment p1-p2.

Return the rotated point. Positive angles are anticlockwise looking down the axis towards the origin. Assume right hand coordinate system.

6.22.3.2 void ComputeCentroid (vtkPolyData * *shape,* double * *centroid*)

compute centroid of polydata

6.22.3.3 int main (int *argc,* char ** *argv*)

6.22.3.4 vtkPolyData * OrderPoints (vtkPolyData * *pts,* int *apical_id,* int *basal_id*)

Something about point order for this specific application.

6.22.3.5 int struct_cmp_by_value (const void * *a,* const void * *b*)

6.23 /home/costa/Data/Code/C/apps/PassScalars.cpp File Reference

copies scalars from one polydata to another

```
#include "vtkPolyDataReader.h"
#include "vtkMath.h"
#include "vtkPointData.h"
#include "vtkPointLocator.h"
#include "vtkPolyData.h"
#include "vtkPolyDataWriter.h"
#include "vtkShortArray.h"
#include "vtkType.h"
#include "vtkCellData.h"
#include "vtkCellLocator.h"
#include "vtkSmartPointer.h"
#include "vtkCell.h"
#include "CommonTools.h"
```

Functions

- int main (int argc, char ∗argv[])

6.23.1 Detailed Description

copies scalars from one polydata to another

6.23.2 Function Documentation

6.23.2.1 int main (int *argc,* char ∗ *argv[]*)

6.24 /home/costa/Data/Code/C/apps/PassScalarsInterp.cpp File Reference

Copies scalars from one mesh to another. When the source is sparse, for every target point it finds a corresponding cell and interpolates the point scalars.

```
#include "vtkPolyDataReader.h"
#include "vtkMath.h"
#include "vtkPointData.h"
#include "vtkPointLocator.h"
#include "vtkPolyData.h"
#include "vtkPolyDataWriter.h"
#include "vtkShortArray.h"
#include "vtkType.h"
#include "vtkCellData.h"
#include "vtkCellLocator.h"
#include "vtkSmartPointer.h"
#include "vtkCell.h"
#include "CommonTools.h"
#include "vtkFloatArray.h"
```

Functions

- void PassScalarsFloat (int argc, char ∗argv[])
- int main (int argc, char ∗argv[])

6.24.1 Detailed Description

Copies scalars from one mesh to another. When the source is sparse, for every target point it finds a corresponding cell and interpolates the point scalars.

6.24.2 Function Documentation

6.24.2.1 int main (int *argc,* char ∗ *argv[]*)

6.24.2.2 void PassScalarsFloat (int *argc,* char ∗ *argv[]*)

6.25 /home/costa/Data/Code/C/apps/PassScalarsReverse.cpp File Reference

Same as PassScalars, except that the search is done for every point/cell of target, not source!

```
#include "vtkPolyDataReader.h"
#include "vtkMath.h"
#include "vtkPointData.h"
#include "vtkPointLocator.h"
#include "vtkPolyData.h"
#include "vtkPolyDataWriter.h"
#include "vtkShortArray.h"
#include "vtkType.h"
#include "vtkCellData.h"
#include "vtkCellLocator.h"
#include "vtkSmartPointer.h"
#include "vtkCell.h"
#include "CommonTools.h"
#include "vtkFloatArray.h"
```

Functions

- void PassScalarsFloat (int argc, char ∗argv[])
- void PassScalarsShort (int argc, char ∗argv[])
- int main (int argc, char ∗argv[])

6.25.1 Detailed Description

Same as PassScalars, except that the search is done for every point/cell of target, not source!

6.25.2 Function Documentation

6.25.2.1 int main (int *argc,* char ∗ *argv[]*)

6.25.2.2 void PassScalarsFloat (int *argc,* char ∗ *argv[]*)

6.25.2.3 void PassScalarsShort (int *argc,* char ∗ *argv[]*)

6.26 /home/costa/Data/Code/C/apps/ResampleImage.cpp File Reference

Resample image, smooth, reconstruct shape.

```
#include <vtkImageData.h>
#include <vtkSmartPointer.h>
#include <vtkImageResample.h>
#include <vtkImageMarchingCubes.h>
#include <vtkImageGaussianSmooth.h>
#include <vtkCleanPolyData.h>
#include <vtkDataSetSurfaceFilter.h>
#include <vtkPolyDataNormals.h>
#include <vtkImageCast.h>
#include <vtkImageConstantPad.h>
#include <vtkImageChangeInformation.h>
#include <vtkShortArray.h>
#include <vtkCellData.h>
#include "CommonTools.h"
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
```

Functions

- int main (int argc, char ∗∗argv)

6.26.1 Detailed Description

Resample image, smooth, reconstruct shape.

6.26.2 Function Documentation

6.26.2.1 int main (int *argc*, char ∗∗ *argv*)

6.27 /home/costa/Data/Code/C/apps/SetScalars.cpp File Reference

adds scalar array ro polydata and sets its value to a constant

```
#include <vtkPointData.h>
#include <vtkShortArray.h>
#include <vtkPolyData.h>
#include <vtkPolyDataWriter.h>
#include <vtkSTLWriter.h>
#include <vtkPolyDataReader.h>
#include <vtkSTLReader.h>
#include "vtkCellData.h"
#include "CommonTools.h"
#include "vtkSmartPointer.h"
#include <vtkDataArray.h>
#include <vtkFloatArray.h>
```

Functions

- template<typename vtk_array_type >
  void AddArray (vtkPolyData ∗shapePt, bool use_points, double value, const char ∗property_name, vtk_array_type ∗fakeparam)
- int main (int argc, char ∗argv[ ])

6.27.1 Detailed Description

adds scalar array ro polydata and sets its value to a constant

6.27.2 Function Documentation

6.27.2.1 template<typename vtk_array_type > void AddArray (vtkPolyData ∗ *shapePt,* bool *use_points,* double *value,* const char ∗ *property_name,* vtk_array_type ∗ *fakeparam*)

6.27.2.2 int main (int *argc,* char ∗ *argv[ ]*)

6.28 /home/costa/Data/Code/C/apps/SmoothMeshTrhoughImage.cpp File Reference

Smooth mesh using image mask.

```
#include "vtkPolyData.h"
#include "vtkCleanPolyData.h"
#include "CommonTools.h"
#include <vtkPolyDataToImageStencil.h>
#include <vtkImageStencil.h>
#include <vtkImageMarchingCubes.h>
#include <vtkImageGaussianSmooth.h>
#include <vtkDataSetWriter.h>
#include <vtkSmartPointer.h>
#include <vtkPolyDataNormals.h>
#include <iostream>
#include <fstream>
#include <stdio.h>
#include "vtkDataSetSurfaceFilter.h"
```

Functions

- int main (int argc, char ∗argv[ ])

6.28.1 Detailed Description

Smooth mesh using image mask.

6.28.2 Function Documentation

6.28.2.1 int main (int *argc,* char ∗ *argv[ ]*)

6.29 /home/costa/Data/Code/C/apps/TransformPhilipsHeart.cpp File Reference

Transforms Philips mesh to suit EM simulations. Don't remember WTF is that.

```
#include "CommonTools.h"
#include <vtkSmartPointer.h>
#include <vtkPolyData.h>
#include <vtkBooleanOperationPolyDataFilter.h>
#include <vtkCleanPolyData.h>
```

Functions

- int main (int argc, char ∗argv[])

6.29.1 Detailed Description

Transforms Philips mesh to suit EM simulations. Don't remember WTF is that.

6.29.2 Function Documentation

6.29.2.1 int main (int *argc,* char ∗ *argv[ ]*)

6.30 /home/costa/Data/Code/C/apps/VTKConvert.cpp File Reference

Convert vtk polydata between formats.

```
#include "CommonTools.h"
#include <vtkSmartPointer.h>
#include <vtkPolyData.h>
```

Functions

- int main (int argc, char ∗argv[])

6.30.1 Detailed Description

Convert vtk polydata between formats.

6.30.2 Function Documentation

6.30.2.1 int main (int *argc,* char ∗ *argv[ ]*)

Index

∼BinaryThinningImageFilter3D
    itk::BinaryThinningImageFilter3D, 18
∼ImageToVTKImageFilter
    itk::ImageToVTKImageFilter, 23
/home/costa/Data/Code/C/apps/Blob.cpp, 29
/home/costa/Data/Code/C/apps/CloseBVMesh.cpp, 31
/home/costa/Data/Code/C/apps/CommonTools.cpp, 33
/home/costa/Data/Code/C/apps/CommonTools.h, 34
/home/costa/Data/Code/C/apps/CreateImageMask.cpp, 36
/home/costa/Data/Code/C/apps/ExtractShapeRegion.cpp, 36
/home/costa/Data/Code/C/apps/GenerateVolumetricLVMesh.cpp, 37
/home/costa/Data/Code/C/apps/LabelBiventricularMesh.cpp, 61
/home/costa/Data/Code/C/apps/MRIRemesh.cpp, 64
/home/costa/Data/Code/C/apps/MakeBiventricularMesh.cpp, 62
/home/costa/Data/Code/C/apps/MeshHeart.cpp, 63
/home/costa/Data/Code/C/apps/MeshSegmentationLaplace.cpp, 64
/home/costa/Data/Code/C/apps/PassScalars.cpp, 66
/home/costa/Data/Code/C/apps/PassScalarsInterp.cpp, 67
/home/costa/Data/Code/C/apps/PassScalarsReverse.cpp, 68
/home/costa/Data/Code/C/apps/ResampleImage.cpp, 68
/home/costa/Data/Code/C/apps/SetScalars.cpp, 69
/home/costa/Data/Code/C/apps/SmoothMeshTrhoughImage.cpp, 70
/home/costa/Data/Code/C/apps/TransformPhilipsHeart.cpp, 70
/home/costa/Data/Code/C/apps/VTKConvert.cpp, 71
/home/costa/Data/Code/C/apps/chaste2vtk.cpp, 30
/home/costa/Data/Code/C/apps/imported/LabelBranches3D.cpp, 40
/home/costa/Data/Code/C/apps/imported/itkBinaryThinningImageFilter3D.h, 39
/home/costa/Data/Code/C/apps/imported/itkImageToVTKImageFilter.h, 40
/home/costa/Data/Code/C/apps/imported/pgm2itkvol/itkvol2pgm.cxx, 42
/home/costa/Data/Code/C/apps/imported/pgm2itkvol/mccodimage.h, 43
/home/costa/Data/Code/C/apps/imported/pgm2itkvol/mcimage.c, 50
/home/costa/Data/Code/C/apps/imported/pgm2itkvol/mcimage.h, 55
/home/costa/Data/Code/C/apps/imported/pgm2itkvol/mcutil.h, 59
/home/costa/Data/Code/C/apps/imported/pgm2itkvol/pgm2itkvol.cxx, 61
__pink__inline
    mcimage.h, 57

ACCEPTED_TYPES1
    mccodimage.h, 45
ACCEPTED_TYPES2
    mccodimage.h, 45
ACCEPTED_TYPES3
    mccodimage.h, 45
ACCEPTED_TYPES4
    mccodimage.h, 45
ACCEPTED_TYPES5
    mccodimage.h, 46
ACCEPTED_TYPES6
    mccodimage.h, 46
ACCEPTED_TYPES7
    mccodimage.h, 46
AddArray
    SetScalars.cpp, 70
addtest
    itk::ImageToVTKImageFilter, 23
addvector
    itk::ImageToVTKImageFilter, 23
allocheader
    mcimage.c, 53
    mcimage.h, 57
allocimage
    mcimage.c, 53
    mcimage.h, 57
allocmultimage
    mcimage.c, 53
    mcimage.h, 57
ApproximateCurvature
    Blob.cpp, 29
ArbitraryRotate
    MRIRemesh.cpp, 66
arrondi
    mcutil.h, 60

BITMAPFILEHEADER, 19
    DataOffset, 19
    FileSize, 19
    reserved, 19
    Signature, 20

BITMAPINFOHEADER, 20
    BitCount, 20
    ColorsImportant, 20
    ColorsUsed, 20
    Compression, 20
    Height, 20
    ImageSize, 20
    Planes, 20
    Size, 20
    Width, 20
    XpixelsPerM, 20
    YpixelsPerM, 20
BUFFERSIZE
    mcimage.c, 52
BinaryThinningImageFilter3D
    itk::BinaryThinningImageFilter3D, 18
BitCount
    BITMAPINFOHEADER, 20
Blob.cpp
    ApproximateCurvature, 29
    GetPointNeighbors, 30
    main, 30
bord
    mccodimage.h, 48
bord3d
    mccodimage.h, 48
bytespercha
    RGBFILEHEADER, 25

COMPARE_SIZE
    mccodimage.h, 46
COMPLEXDATA
    mccodimage.h, 46
CONTOUR_ENDO
    MRIRemesh.cpp, 66
CONTOUR_ENDO_BASAL
    MRIRemesh.cpp, 66
CONTOUR_EPI
    MRIRemesh.cpp, 66
CONTOUR_EPI_APICAL
    MRIRemesh.cpp, 66
CONTOUR_EPI_BASAL
    MRIRemesh.cpp, 66
chaste2vtk.cpp
    main, 30
    ReadChasteElements, 30
    ReadChasteNodes, 30
check_neighborhood
    LabelBranches3D.cpp, 41
check_num_connected_neighbors
    LabelBranches3D.cpp, 41
CheckConnect
    LabelBranches3D.cpp, 41
CheckPointState
    LabelBranches3D.cpp, 41
CheckSaveFileExtension
    CommonTools, 9
CloseBVMesh.cpp
    CopyCellScalars, 31
    FillSmallHoles, 31
    main, 31
    PointInPolygon, 32
    PolyData2Polygon, 32
    PolygonBoundaryArea, 32
CloseSurface
    CommonTools, 9
cmaptype
    RGBFILEHEADER, 25
col_size
    xvimage, 27
ColorsImportant
    BITMAPINFOHEADER, 20
ColorsUsed
    BITMAPINFOHEADER, 20
colsize
    mccodimage.h, 46
CommonTools
    PLYType, 9
    STLType, 9
    UnknownType, 9
    UnknownVolumeType, 9
    VTKPolyDataType, 9
    VTKUnstructuredGridType, 9
    VTKXMLPolyDataType, 9
CommonTools, 7
    CheckSaveFileExtension, 9
    CloseSurface, 9
    ExportPolyDataPoints, 9
    FileExists, 9
    GenerateDecimationScript, 9
    GenerateHoleCover, 10
    GetP2P, 10
    GetP2S, 10
    GetS2S, 10
    GetShapeSubSurface, 10
    GetTypeOfVTKData, 10
    GetTypeOfVTKVolumeData, 10
    ImportPolyDataPoints, 11
    laplace3D_voxelsize, 11
    LoadImage, 11
    LoadShapeFromFile, 11
    LoadVolumeFromFile, 11
    LoadVtkShortArray, 11
    Points2Polydata, 11
    ReadFilelist, 12
    SaveImage, 12
    SavePoints, 12
    SavePolydata, 12
    SaveShapeToFile, 12
    SaveUnstructuredGrid, 12
    SaveVolumeToFile, 12
    SaveVtkShortArray, 12
    ScaleShape, 12
    ScaleVolume, 13
    ShrinkImage, 13
    VTKSurfaceMeshFormats, 9
    VTKVolumeMeshFormats, 9

comp_conexas
LabelBranches3D.cpp, 41
components
RGBFILEHEADER, 25
Compression
BITMAPINFOHEADER, 20
compression
RGBFILEHEADER, 25
ComputeCentroid
MRIRemesh.cpp, 66
ComputeThinImage
itk::BinaryThinningImageFilter3D, 18
ConstBoundaryConditionType
itk::BinaryThinningImageFilter3D, 16
ConstPointer
itk::BinaryThinningImageFilter3D, 16
itk::ImageToVTKImageFilter, 23
convertfloat
mcimage.c, 53
mcimage.h, 57
convertgen
mcimage.c, 53
mcimage.h, 57
convertlong
mcimage.c, 53
mcimage.h, 57
copy2image
mcimage.c, 53
mcimage.h, 57
CopyCellScalars
CloseBVMesh.cpp, 31
copyimage
mcimage.c, 53
mcimage.h, 57
CreateEmptyImage
MakeBiventricularMesh.cpp, 63
CreateImageMask.cpp
main, 36
CreateMask
MakeBiventricularMesh.cpp, 63

d
xvimage, 27
DCOMPLEXDATA
mccodimage.h, 47
DERRIERE
mccodimage.h, 47
DEVANT
mccodimage.h, 47
DOUBLEDATA
mccodimage.h, 47
data_storage_type
xvimage, 27
DataOffset
BITMAPFILEHEADER, 19
datatype
mccodimage.h, 47
dcomplex, 21
im, 21
re, 21
debug
LabelBranches3D.cpp, 42
DecimateMesh
MakeBiventricularMesh.cpp, 63
depth
mccodimage.h, 47
depth_size
xvimage, 27
dim
RGBFILEHEADER, 25
distance
LabelBranches3D.cpp, 41
dummy
RGBFILEHEADER, 25

EST
mccodimage.h, 47
equalimages
mcimage.c, 53
mcimage.h, 57
ExportPolyDataPoints
CommonTools, 9
ExporterFilterPointer
itk::ImageToVTKImageFilter, 23
ExporterFilterType
itk::ImageToVTKImageFilter, 23
ExtractShapeRegion.cpp
main, 37
ExtractSurface
MeshHeart.cpp, 64

F_NAME
mcimage.c, 52, 53
FIELD_DT
GenerateVolumetricLVMesh.cpp, 39
FIELD_LAPLACE
GenerateVolumetricLVMesh.cpp, 39
FLOATDATA
mccodimage.h, 47
FLT_MAX
LabelBranches3D.cpp, 41
fcomplex, 21
im, 22
re, 22
FileExists
CommonTools, 9
FileSize
BITMAPFILEHEADER, 19
fillEulerLUT
itk::BinaryThinningImageFilter3D, 18
FillSmallHoles
CloseBVMesh.cpp, 31
find_maximun
LabelBranches3D.cpp, 41
find_shorter_branch
LabelBranches3D.cpp, 41
FindTriplePoints
LabelBranches3D.cpp, 42

findmaximum_centerline
LabelBranches3D.cpp, 42
freadulong
mcimage.c, 53
freadushort
mcimage.c, 53
freeimage
mcimage.c, 53
mcimage.h, 57
fwriteulong
mcimage.c, 54
fwriteushort
mcimage.c, 54

GenerateData
itk::BinaryThinningImageFilter3D, 18
GenerateDecimationScript
CommonTools, 9
GenerateHoleCover
CommonTools, 10
GenerateImageMask
GenerateVolumetricLVMesh.cpp, 39
GenerateLayersAlongField
GenerateVolumetricLVMesh.cpp, 39
GenerateLayersAlongNormals
GenerateVolumetricLVMesh.cpp, 39
GenerateLocalCoordinateCircLongit
GenerateVolumetricLVMesh.cpp, 39
GenerateLocalCoordinateRadial
GenerateVolumetricLVMesh.cpp, 39
GenerateVolumetricLVMesh.cpp
FIELD_DT, 39
FIELD_LAPLACE, 39
GenerateImageMask, 39
GenerateLayersAlongField, 39
GenerateLayersAlongNormals, 39
GenerateLocalCoordinateCircLongit, 39
GenerateLocalCoordinateRadial, 39
main, 39
usage, 39
GetExporter
itk::ImageToVTKImageFilter, 24
GetImporter
itk::ImageToVTKImageFilter, 24
GetOutput
itk::ImageToVTKImageFilter, 24
GetP2P
CommonTools, 10
GetP2S
CommonTools, 10
GetPointNeighbors
Blob.cpp, 30
GetS2S
CommonTools, 10
GetShapeSubSurface
CommonTools, 10
GetThinning
itk::BinaryThinningImageFilter3D, 18
GetTypeOfVTKData
CommonTools, 10
GetTypeOfVTKVolumeData
CommonTools, 10
getvtest
itk::ImageToVTKImageFilter, 24

HUGE_IMAGE_SIZE
mcimage.h, 57
Height
BITMAPINFOHEADER, 20
height
RGBFILEHEADER, 25

INT32_MAX
mcimage.c, 53
im
dcomplex, 21
fcomplex, 22
image2list
mcimage.c, 54
mcimage.h, 58
image_data
xvimage, 27
ImageSize
BITMAPINFOHEADER, 20
ImageToVTKImageFilter
itk::ImageToVTKImageFilter, 23
ImportPolyDataPoints
CommonTools, 11
index
sorting_struct, 26
index_t
mcimage.h, 57
IndexType
itk::BinaryThinningImageFilter3D, 16
InputImagePixelType
itk::BinaryThinningImageFilter3D, 16
InputImagePointer
itk::BinaryThinningImageFilter3D, 17
itk::ImageToVTKImageFilter, 23
InputImageType
itk::BinaryThinningImageFilter3D, 17
itk::ImageToVTKImageFilter, 23
int32_t
mcimage.h, 57
isEulerInvariant
itk::BinaryThinningImageFilter3D, 18
isSimplePoint
itk::BinaryThinningImageFilter3D, 18
itk, 13
itk::BinaryThinningImageFilter3D
∼BinaryThinningImageFilter3D, 18
BinaryThinningImageFilter3D, 18
ComputeThinImage, 18
ConstBoundaryConditionType, 16
ConstPointer, 16
fillEulerLUT, 18
GenerateData, 18
GetThinning, 18

IndexType, 16
InputImagePixelType, 16
InputImagePointer, 17
InputImageType, 17
isEulerInvariant, 18
isSimplePoint, 18
itkNewMacro, 18
itkStaticConstMacro, 19
itkTypeMacro, 19
NeighborhoodIteratorType, 17
NeighborhoodType, 17
Octree_labeling, 19
OutputImagePixelType, 17
OutputImagePointer, 17
OutputImageType, 17
Pointer, 17
PrepareData, 19
PrintSelf, 19
RegionType, 17
Self, 17
SizeType, 18
Superclass, 18
itk::BinaryThinningImageFilter3D< TInputImage, TOutputImage >, 15
itk::ImageToVTKImageFilter
~ImageToVTKImageFilter, 23
addtest, 23
addvector, 23
ConstPointer, 23
ExporterFilterPointer, 23
ExporterFilterType, 23
GetExporter, 24
GetImporter, 24
GetOutput, 24
getvtest, 24
ImageToVTKImageFilter, 23
InputImagePointer, 23
InputImageType, 23
itkNewMacro, 24
itkTypeMacro, 24
Pointer, 23
Self, 23
SetInput, 24
Superclass, 23
testsize, 24
tralala, 24
Update, 24
itk::ImageToVTKImageFilter< TInputImage >, 22
itkNewMacro
itk::BinaryThinningImageFilter3D, 18
itk::ImageToVTKImageFilter, 24
itkStaticConstMacro
itk::BinaryThinningImageFilter3D, 19
itkTypeMacro
itk::BinaryThinningImageFilter3D, 19
itk::ImageToVTKImageFilter, 24
itkvol2pgm.cxx
main, 43
VERBOSE, 43

LabelBiventricularMesh.cpp
main, 62
SetScalars, 62
LabelBranches3D.cpp
check_neighborhood, 41
check_num_connected_neighbors, 41
CheckConnect, 41
CheckPointState, 41
comp_conexas, 41
debug, 42
distance, 41
FLT_MAX, 41
find_maximun, 41
find_shorter_branch, 41
FindTriplePoints, 42
findmaximum_centerline, 42
LabelBranchs, 42
LabelBranchsNew, 42
main, 42
mapIndex3D, 42
maptox, 42
maptoy, 42
maptoz, 42
print_timing, 42
Relabeling, 42
reserve_memory_triple_float, 42
reserve_memory_triple_int, 42
verbose, 42
LabelBranchs
LabelBranches3D.cpp, 42
LabelBranchsNew
LabelBranches3D.cpp, 42
laplace3D_voxelsize
CommonTools, 11
list2image
mcimage.c, 54
mcimage.h, 58
LoadImage
CommonTools, 11
LoadShapeFromFile
CommonTools, 11
LoadVolumeFromFile
CommonTools, 11
LoadVtkShortArray
CommonTools, 11
lpixel
mccodimage.h, 47
lvoxel
mccodimage.h, 47

M_1_PI
mcutil.h, 60
M_2_PI
mcutil.h, 60
M_2_SQRTPI
mcutil.h, 60
M_E

mcutil.h, 60
M_LN10
mcutil.h, 60
M_LN2
mcutil.h, 60
M_LOG10E
mcutil.h, 60
M_LOG2E
mcutil.h, 60
M_PI
mcutil.h, 60
M_PI_2
mcutil.h, 60
M_PI_4
mcutil.h, 60
M_SQRT1_2
mcutil.h, 60
M_SQRT2
mcutil.h, 60
MRIRemesh.cpp
ArbitraryRotate, 66
CONTOUR_ENDO, 66
CONTOUR_EPI, 66
ComputeCentroid, 66
main, 66
OrderPoints, 66
REF_POINT, 66
struct_cmp_by_value, 66
magic
RGBFILEHEADER, 25
main
Blob.cpp, 30
chaste2vtk.cpp, 30
CloseBVMesh.cpp, 31
CreateImageMask.cpp, 36
ExtractShapeRegion.cpp, 37
GenerateVolumetricLVMesh.cpp, 39
itkvol2pgm.cxx, 43
LabelBiventricularMesh.cpp, 62
LabelBranches3D.cpp, 42
MakeBiventricularMesh.cpp, 63
MeshHeart.cpp, 64
MeshSegmentationLaplace.cpp, 64
MRIRemesh.cpp, 66
PassScalars.cpp, 67
PassScalarsInterp.cpp, 67
PassScalarsReverse.cpp, 68
pgm2itkvol.cxx, 61
ResampleImage.cpp, 69
SetScalars.cpp, 70
SmoothMeshTrhoughImage.cpp, 70
TransformPhilipsHeart.cpp, 71
VTKConvert.cpp, 71
MakeBiventricularMesh.cpp
CreateEmptyImage, 63
CreateMask, 63
DecimateMesh, 63
main, 63
mapIndex3D
LabelBranches3D.cpp, 42
maptox
LabelBranches3D.cpp, 42
maptoy
LabelBranches3D.cpp, 42
maptoz
LabelBranches3D.cpp, 42
maskvois26
mccodimage.h, 48
maxcol
RGBFILEHEADER, 25
mcabs
mcutil.h, 60
mccodimage.h
ACCEPTED_TYPES1, 45
ACCEPTED_TYPES2, 45
ACCEPTED_TYPES3, 45
ACCEPTED_TYPES4, 45
ACCEPTED_TYPES5, 46
ACCEPTED_TYPES6, 46
ACCEPTED_TYPES7, 46
bord, 48
bord3d, 48
COMPARE_SIZE, 46
COMPLEXDATA, 46
colsize, 46
DCOMPLEXDATA, 47
DERRIERE, 47
DEVANT, 47
DOUBLEDATA, 47
datatype, 47
depth, 47
EST, 47
FLOATDATA, 47
lpixel, 47
lvoxel, 47
maskvois26, 48
NDG_MAX, 47
NDG_MIN, 47
NORD, 47
NORD_EST, 47
NORD_OUEST, 47
nbands, 47
nonbord, 47
nonbord3d, 47
ONLY_2D, 47
ONLY_3D, 47
OUEST, 47
pixel, 48
rowsize, 48
SCHARDATA, 48
SLONGDATA, 48
SSHORTDATA, 48
SUD, 48
SUD_EST, 48
SUD_OUEST, 48
sont18voisins, 48

sont26voisins, 49
sont4voisins, 49
sont6voisins, 49
sont8voisins, 49
tsize, 48
UCHARDATA, 48
ULONGDATA, 48
USHORTDATA, 48
VFF_TYP_1_BYTE, 48
VFF_TYP_2_BYTE, 48
VFF_TYP_4_BYTE, 48
VFF_TYP_BIT, 48
VFF_TYP_COMPLEX, 48
VFF_TYP_DCOMPLEX, 48
VFF_TYP_DOUBLE, 48
VFF_TYP_FLOAT, 48
voisin, 49
voisin125, 49
voisin14b, 49
voisin18, 49
voisin2, 49
voisin26, 49
voisin5, 49
voisin6, 49
voisin6b, 49
voisinENAR, 49
voisinENAV, 49
voisinNESO, 49
voisinNOSE, 49
voisinONAR, 49
voisinONAV, 49
voisins18, 49
voisins26, 49
voisins4, 49
voisins6, 49
voisins8, 49
voxel, 48
xvimage, 48

mceven
mcutil.h, 60

mcimage.c
allocheader, 53
allocimage, 53
allocmultimage, 53
BUFFERSIZE, 52
convertfloat, 53
convertgen, 53
convertlong, 53
copy2image, 53
copyimage, 53
equalimages, 53
F_NAME, 52, 53
freadulong, 53
freadushort, 53
freeimage, 53
fwriteulong, 54
fwriteushort, 54
INT32_MAX, 53
image2list, 54
list2image, 54
pink_fopen_read, 54
pink_fopen_write, 54
printimage, 54
razimage, 54
readbmp, 54
readheader, 54
readimage, 54
readlongimage, 54
readrgb, 54
readrgbimage, 54
readse, 55
showheader, 55
WARN_HUGE, 53
writeascimage, 55
writebmp, 55
writeimage, 55
writelist2, 55
writelist3, 55
writelongimage, 55
writerawimage, 55
writergbascimage, 55
writergbimage, 55
writese, 55

mcimage.h
__pink__inline, 57
allocheader, 57
allocimage, 57
allocmultimage, 57
convertfloat, 57
convertgen, 57
convertlong, 57
copy2image, 57
copyimage, 57
equalimages, 57
freeimage, 57
HUGE_IMAGE_SIZE, 57
image2list, 58
index_t, 57
int32_t, 57
list2image, 58
printimage, 58
razimage, 58
readbmp, 58
readheader, 58
readimage, 58
readlongimage, 58
readrgb, 58
readrgbimage, 58
readse, 58
showheader, 59
u_int32_t, 57
u_int8_t, 57
uint32_t, 57
uint8_t, 57
writeascimage, 59
writebmp, 59

writeimage, 59
writelist2, 59
writelist3, 59
writelongimage, 59
writerawimage, 59
writergbascimage, 59
writergbimage, 59
writese, 59
mcmax
mcutil.h, 60
mcmin
mcutil.h, 60
mcodd
mcutil.h, 60
mcsqr
mcutil.h, 60
mcutil.h
arrondi, 60
M_1_PI, 60
M_2_PI, 60
M_2_SQRTPI, 60
M_E, 60
M_LN10, 60
M_LN2, 60
M_LOG10E, 60
M_LOG2E, 60
M_PI, 60
M_PI_2, 60
M_PI_4, 60
M_SQRT1_2, 60
M_SQRT2, 60
mcabs, 60
mceven, 60
mcmax, 60
mcmin, 60
mcodd, 60
mcsqr, 60
signe, 60
MeshHeart.cpp
ExtractSurface, 64
main, 64
MeshSegmentationLaplace.cpp
main, 64
mincol
RGBFILEHEADER, 25

NDG_MAX
mccodimage.h, 47
NDG_MIN
mccodimage.h, 47
NORD
mccodimage.h, 47
NORD_EST
mccodimage.h, 47
NORD_OUEST
mccodimage.h, 47
name
RGBFILEHEADER, 25
xvimage, 27
nbands
mccodimage.h, 47
NeighborhoodIteratorType
itk::BinaryThinningImageFilter3D, 17
NeighborhoodType
itk::BinaryThinningImageFilter3D, 17
nonbord
mccodimage.h, 47
nonbord3d
mccodimage.h, 47
num_data_bands
xvimage, 27

ONLY_2D
mccodimage.h, 47
ONLY_3D
mccodimage.h, 47
OUEST
mccodimage.h, 47
Octree_labeling
itk::BinaryThinningImageFilter3D, 19
OrderPoints
MRIRemesh.cpp, 66
origin_x
xvimage, 27
origin_y
xvimage, 27
origin_z
xvimage, 27
OutputImagePixelType
itk::BinaryThinningImageFilter3D, 17
OutputImagePointer
itk::BinaryThinningImageFilter3D, 17
OutputImageType
itk::BinaryThinningImageFilter3D, 17

PLYType
CommonTools, 9
PassScalars.cpp
main, 67
PassScalarsFloat
PassScalarsInterp.cpp, 67
PassScalarsReverse.cpp, 68
PassScalarsInterp.cpp
main, 67
PassScalarsFloat, 67
PassScalarsReverse.cpp
main, 68
PassScalarsFloat, 68
PassScalarsShort, 68
PassScalarsShort
PassScalarsReverse.cpp, 68
pgm2itkvol.cxx
main, 61
UINTDATA, 61
VERBOSE, 61
pink_fopen_read
mcimage.c, 54
pink_fopen_write

mcimage.c, 54
pixel
mccodimage.h, 48
Planes
BITMAPINFOHEADER, 20
PointInPolygon
CloseBVMesh.cpp, 32
Pointer
itk::BinaryThinningImageFilter3D, 17
itk::ImageToVTKImageFilter, 23
Points2Polydata
CommonTools, 11
PolyData2Polygon
CloseBVMesh.cpp, 32
PolygonBoundaryArea
CloseBVMesh.cpp, 32
PrepareData
itk::BinaryThinningImageFilter3D, 19
print_timing
LabelBranches3D.cpp, 42
PrintSelf
itk::BinaryThinningImageFilter3D, 19
printimage
mcimage.c, 54
mcimage.h, 58

REF_POINT
MRIRemesh.cpp, 66
RGBFILEHEADER, 25
bytespercha, 25
cmaptype, 25
components, 25
compression, 25
dim, 25
dummy, 25
height, 25
magic, 25
maxcol, 25
mincol, 25
name, 25
width, 25
razimage
mcimage.c, 54
mcimage.h, 58
re
dcomplex, 21
fcomplex, 22
ReadChasteElements
chaste2vtk.cpp, 30
ReadChasteNodes
chaste2vtk.cpp, 30
ReadFilelist
CommonTools, 12
readbmp
mcimage.c, 54
mcimage.h, 58
readheader
mcimage.c, 54
mcimage.h, 58
readimage
mcimage.c, 54
mcimage.h, 58
readlongimage
mcimage.c, 54
mcimage.h, 58
readrgb
mcimage.c, 54
mcimage.h, 58
readrgbimage
mcimage.c, 54
mcimage.h, 58
readse
mcimage.c, 55
mcimage.h, 58
RegionType
itk::BinaryThinningImageFilter3D, 17
Relabeling
LabelBranches3D.cpp, 42
ResampleImage.cpp
main, 69
reserve_memory_triple_float
LabelBranches3D.cpp, 42
reserve_memory_triple_int
LabelBranches3D.cpp, 42
reserved
BITMAPFILEHEADER, 19
row_size
xvimage, 27
rowsize
mccodimage.h, 48

STLType
CommonTools, 9
SCHARDATA
mccodimage.h, 48
SLONGDATA
mccodimage.h, 48
SSHORTDATA
mccodimage.h, 48
SUD
mccodimage.h, 48
SUD_EST
mccodimage.h, 48
SUD_OUEST
mccodimage.h, 48
SaveImage
CommonTools, 12
SavePoints
CommonTools, 12
SavePolydata
CommonTools, 12
SaveShapeToFile
CommonTools, 12
SaveUnstructuredGrid
CommonTools, 12
SaveVolumeToFile
CommonTools, 12
SaveVtkShortArray

CommonTools, 12
ScaleShape
CommonTools, 12
ScaleVolume
CommonTools, 13
Self
itk::BinaryThinningImageFilter3D, 17
itk::ImageToVTKImageFilter, 23
SetInput
itk::ImageToVTKImageFilter, 24
SetScalars
LabelBiventricularMesh.cpp, 62
SetScalars.cpp
AddArray, 70
main, 70
showheader
mcimage.c, 55
mcimage.h, 59
ShrinkImage
CommonTools, 13
Signature
BITMAPFILEHEADER, 20
signe
mcutil.h, 60
Size
BITMAPINFOHEADER, 20
SizeType
itk::BinaryThinningImageFilter3D, 18
SmoothMeshTrhoughImage.cpp
main, 70
sont18voisins
mccodimage.h, 48
sont26voisins
mccodimage.h, 49
sont4voisins
mccodimage.h, 49
sont6voisins
mccodimage.h, 49
sont8voisins
mccodimage.h, 49
sorting_struct, 25
index, 26
value, 26
struct_cmp_by_value
MRIRemesh.cpp, 66
Superclass
itk::BinaryThinningImageFilter3D, 18
itk::ImageToVTKImageFilter, 23

testsize
itk::ImageToVTKImageFilter, 24
time_size
xvimage, 27
tralala
itk::ImageToVTKImageFilter, 24
TransformPhilipsHeart.cpp
main, 71
tsize
mccodimage.h, 48

u_int32_t
mcimage.h, 57
u_int8_t
mcimage.h, 57
UCHARDATA
mccodimage.h, 48
UINTDATA
pgm2itkvol.cxx, 61
ULONGDATA
mccodimage.h, 48
USHORTDATA
mccodimage.h, 48
uint32_t
mcimage.h, 57
uint8_t
mcimage.h, 57
UnknownType
CommonTools, 9
UnknownVolumeType
CommonTools, 9
Update
itk::ImageToVTKImageFilter, 24
usage
GenerateVolumetricLVMesh.cpp, 39

VTKPolyDataType
CommonTools, 9
VTKUnstructuredGridType
CommonTools, 9
VTKXMLPolyDataType
CommonTools, 9
VERBOSE
itkvol2pgm.cxx, 43
pgm2itkvol.cxx, 61
VFF_TYP_1_BYTE
mccodimage.h, 48
VFF_TYP_2_BYTE
mccodimage.h, 48
VFF_TYP_4_BYTE
mccodimage.h, 48
VFF_TYP_BIT
mccodimage.h, 48
VFF_TYP_COMPLEX
mccodimage.h, 48
VFF_TYP_DCOMPLEX
mccodimage.h, 48
VFF_TYP_DOUBLE
mccodimage.h, 48
VFF_TYP_FLOAT
mccodimage.h, 48
VTKConvert.cpp
main, 71
VTKSurfaceMeshFormats
CommonTools, 9
VTKVolumeMeshFormats
CommonTools, 9
value
sorting_struct, 26
verbose

LabelBranches3D.cpp, 42
voisin
mccodimage.h, 49
voisin125
mccodimage.h, 49
voisin14b
mccodimage.h, 49
voisin18
mccodimage.h, 49
voisin2
mccodimage.h, 49
voisin26
mccodimage.h, 49
voisin5
mccodimage.h, 49
voisin6
mccodimage.h, 49
voisin6b
mccodimage.h, 49
voisinENAR
mccodimage.h, 49
voisinENAV
mccodimage.h, 49
voisinNESO
mccodimage.h, 49
voisinNOSE
mccodimage.h, 49
voisinONAR
mccodimage.h, 49
voisinONAV
mccodimage.h, 49
voisins18
mccodimage.h, 49
voisins26
mccodimage.h, 49
voisins4
mccodimage.h, 49
voisins6
mccodimage.h, 49
voisins8
mccodimage.h, 49
voxel
mccodimage.h, 48

WARN_HUGE
mcimage.c, 53
Width
BITMAPINFOHEADER, 20
width
RGBFILEHEADER, 25
writeascimage
mcimage.c, 55
mcimage.h, 59
writebmp
mcimage.c, 55
mcimage.h, 59
writeimage
mcimage.c, 55
mcimage.h, 59
writelist2
mcimage.c, 55
mcimage.h, 59
writelist3
mcimage.c, 55
mcimage.h, 59
writelongimage
mcimage.c, 55
mcimage.h, 59
writerawimage
mcimage.c, 55
mcimage.h, 59
writergbascimage
mcimage.c, 55
mcimage.h, 59
writergbimage
mcimage.c, 55
mcimage.h, 59
writese
mcimage.c, 55
mcimage.h, 59

xdim
xvimage, 28
xmax
xvimage, 28
xmin
xvimage, 28
XpixelsPerM
BITMAPINFOHEADER, 20
xvimage, 26
col_size, 27
d, 27
data_storage_type, 27
depth_size, 27
image_data, 27
mccodimage.h, 48
name, 27
num_data_bands, 27
origin_x, 27
origin_y, 27
origin_z, 27
row_size, 27
time_size, 27
xdim, 28
xmax, 28
xmin, 28
ydim, 28
ymax, 28
ymin, 28
zdim, 28
zmax, 28
zmin, 28

ydim
xvimage, 28
ymax
xvimage, 28
ymin

xvimage, 28

YpixelsPerM

BITMAPINFOHEADER, 20

zdim

xvimage, 28

zmax

xvimage, 28

zmin

xvimage, 28