Red File: /libs/core/cvCore.reds
Source: core_c.h
Fonctions: 332

Fonction	
cvCreateImageHeader	Allocates and initializes IplImage header
cvInitImageHeader	Inializes IplImage header
cvCreateImage	Creates IPL image (header and data); creates new image
cvReleaseImageHeader	Releases (i.e. deallocates) IPL image header
cvReleaseImage	Releases IPL image header and data
cvCloneImage	The function cvCloneImage makes a full copy of the image including header, ROI and data (widthStep may differ)
cvSetImageCOI	The function cvSetImageCOI sets the channel of interest to a given value. Value 0 means that all channels are selected, 1 means that the first channel is selected etc. If ROI is NULL and coi != 0, ROI is allocated. Note that most of OpenCV functions do not support COI, so to process separate image/matrix channel one may copy (via cvCopy or cvSplit) the channel to separate image/matrix, process it and copy the result back (via cvCopy or cvCvtPlaneToPix) if need.
cvGetImageCOI	The function cvGetImageCOI returns channel of interest of the image (it returns 0 if all the channels are selected)
cvSetImageROI	The function cvSetImageROI sets the image ROI to a given rectangle. If ROI is NULL and the value of the parameter rect is not equal to the whole image, ROI is allocated. Unlike COI, most of OpenCV functions do support ROI and treat it in a way as it would be a separate image (for example, all the pixel coordinates are counted from top-left or bottom-left (depending on the image origin) corner of ROI)
cvResetImageROI	he function cvResetImageROI releases image ROI. After that the whole image is considered selected
cvGetImageROI	The function cvGetImageROI returns image ROI coordinates. The rectangle cvRect(0

	0,image/width image/height) is returned if there is no ROI
cvCreateMatHeader	The function cvCreateMatHeader allocates new matrix header and returns pointer to it. The matrix data can further be allocated using cvCreateData or set explicitly to user-allocated data via cvSetData.
cvInitMatHeader	The function `cvInitMatHeader` initializes already allocated CvMat structure. It can be used to process raw data with OpenCV matrix functions.
cvCreateMat	The function cvCreateMat allocates header for the new matrix and underlying data, and returns a pointer to the created matrix.
cvReleaseMat	The function cvReleaseMat decrements the matrix data reference counter and releases matrix heade
cvDecRefData	Inline function
cvIncRefData	Inline function
cvCloneMat	The function cvCloneMat creates a copy of input matrix (except, may be, step value) and returns the pointer to it.
cvGetSubRect	Makes a new matrix from <rect> subrectangle of input array No data is copied
cvGetRows	Selects row span of the input array: arr(start_row:delta_row:end_row (end_row is not included into the span)
cvGetRow	Inline function
cvGetCols	Selects column span of the input array: arr(:,start_col:end_col) (end_col is not included into the span)
cvGetCol	Inline function
cvGetDiag	The function cvGetDiag returns the header, corresponding to a specified diagonal of the input array.
cvScalarToRawData	low-level scalar <-> raw data conversion functions
cvRawDataToScalar	low-level scalar <-> raw data conversion functions
cvCreateMatNDHeader	The function cvCreateMatND allocates header for multi-dimensional dense array. The array data can further be allocated using cvCreateData or set explicitly to user-allocated data via cvSetData.
cvCreateMatND	The function cvCreateMatND allocates header for multi-dimensional dense array and the

	underlying data, and returns pointer to the created array.
cvInitMatNDHeader	Initializes preallocated CvMatND header
cvReleaseMatND	Release CVMatND
cvCloneMatND	Creates a copy of CvMatND (except, may be, steps)
cvCreateSparseMat	Allocates and initializes CvSparseMat header and allocates data"
cvReleaseSparseMat	Releases CvSparseMat
cvCloneSparseMat	Creates a copy of CvSparseMat (except, may be, zero items)
cvInitSparseMatIterator	Initializes sparse array iterator (returns the first node or NULL if the array is empty)
cvGetNextSparseNode	returns next sparse array node (or NULL if there is no more nodes)
cvInitNArrayIterator	initializes iterator that traverses through several arrays simulteneously
cvNextNArraySlice	returns zero value if iteration is finished, non-zero (slice length) otherwise
cvGetElemType	Returns type of array elements: CV_8UC1 ... CV_64FC4 ...
cvGetDims	Retrieves number of an array dimensions and optionally sizes of the dimensions
cvGetDimSize	Retrieves size of a particular array dimension. ;For 2d arrays cvGetDimSize(arr,0) returns number of rows (image height) and cvGetDimSize(arr,1) returns number of columns (image width)
cvPtr1D, cvPtr2D, cvPtr3D, cvPtrND	All indexes are zero-based, the major dimensions go first (e.g. (y,x) for 2D, (z,y,x)
cvGet1D, cvGet2D, cvGet3D, cvGetND	The functions return a specific array element. Renvoient un scalar: pb
cvGetReal1D, cvGetReal2D, cvGetReal3D, cvGetRealND	Real
cvSet1D, cvSet2D, cvSet3D , cvSetND	arr(idx0,idx1,...) = value 4 decimal! Ex: cvSet1D mat1 0 1.0 2.0 3.0 4.0
cvSetReal1D, cvSetReal2D, cvSetReal3D, cvSetRealND	Value: a decimal only
cvClearND	clears element of ND dense array, in case of sparse arrays it deletes the specified node
cvGetMat	Converts CvArr (IplImage or CvMat,...) to CvMat
cvGetImage	Converts CvArr (IplImage or CvMat) to IplImage
cvReshapeMatND	Changes a shape of multi-dimensional array.
cvReshapeND	Inline function

cvReshape	Changes a shape of array
cvRepeat	Repeats source 2d array several times in both horizontal and vertical direction to fill destination array
cvCreateData	Allocates array data
cvReleaseData	Releases array data
cvSetData	Attaches user data to the array header
cvGetRawData	retrieves raw data of CvMat, IplImage or CvMatND
cvGetSize	Returns width and height of array in elements
cvCopy	copies source array to destination array
cvSet	sets all or masked elements of input array to the same value
cvSetZero	clears all the array elements (sets them to 0)
cvZero	alias cvSetZero cvZero
cvSplit	splits a multi-channel array into the set of single-channel arrays or extracts particular [color] plane
cvMerge	merges a set of single-channel arrays into the single multi-channel array or inserts one particular [color] plane to the array
cvMixChannels	Copies several channels from input arrays to certain channels of output arrays
cvConvertScale	Performs linear transformation on every source array element: dst(x,y,c) = scale*src(x,y,c)+shift. Arbitrary combination of input and output array depths are allowed (number of channels must be the same), thus the function can be used for type conversion
cvCvtScale	idem
cvScale	idem
cvConvert	cvConvertScale 1.0 0.0
cvConvertScaleAbs	Warning: destination array must have 8u type. In other cases one may use cvConvertScale + cvAbsDiffS
cvCvtScaleAbs	idem
cvCheckTermCriteria	checks termination criteria validity
cvAdd	dst(mask) = src1(mask) + src2(mask);
cvAddS	dst(mask) = src(mask) + value
cvSub	dst(mask) = src1(mask) - src2(mask)
cvSubS	dst(mask) = src(mask) - value = src(mask) + (-value)
cvSubRS	dst(mask) = value - src(mask)

cvMul	dst(idx) = src1(idx) * src2(idx) * scale (scaled element-wise multiplication of 2 arrays)
cvDiv	element-wise division/inversion with scaling: dst(idx) = src1(idx) * scale / src2(idx) or dst(idx) = scale / src2(idx) if src1 == 0
cvScaleAdd	dst = src1 * scale + src2 */
cvAXPY	Inline function
cvAddWeighted	dst = src1 * alpha + src2 * beta + gamma
cvDotProduct	result = sum_i(src1(i) * src2(i)) (results for all channels are accumulated together)
cvAnd	dst(idx) = src1(idx) & src2(idx)
cvAndS	dst(idx) = src(idx) & value
cvOr	dst(idx) = src1(idx) | src2(idx)
cvOrS	dst(idx) = src(idx) | value
cvXor	dst(idx) = src1(idx) ^ src2(idx)
cvXorS	dst(idx) = src(idx) ^ value
cvNot	dst(idx) = ~src(idx)
cvInRange	dst(idx) = lower(idx) <= src(idx) < upper(idx)
cvInRangeS	;dst(idx) = lower <= src(idx)
cvCmp	dst(idx) = src1(idx) _cmp_op_ src2(idx)
cvCmpS	dst(idx) = src1(idx) _cmp_op_ value
cvMin	dst(idx) = min(src1(idx),src2(idx)
cvMax	dst(idx) = max(src1(idx),src2(idx))
cvMinS	dst(idx) = min(src(idx),value)
cvMaxS	dst(idx) = max(src(idx),value)
cvAbsDiff	dst(x,y,c) = abs(src1(x,y,c) - src2(x,y,c))
cvAbsDiffS	dst(x,y,c) = abs(src(x,y,c) - value(c))
cvAbs	Inline Alias
cvCartToPolar	Does cartesian->polar coordinates conversion. Either of output components (magnitude or angle) is optional"
cvPolarToCart	{Does polar->cartesian coordinates conversion. Either of output components (magnitude or angle) is optional. If magnitude is missing it is assumed to be all 1's
cvPow	Does powering: dst(idx) = src(idx)^power
cvExp	Does exponention: dst(idx) = exp(src(idx)). Overflow is not handled yet. Underflow is handled. Maximal relative error is ~7e-6 for single-precision input
cvLog	Calculates natural logarithms: dst(idx) = log(abs(src(idx))).

	Logarithm of 0 gives large negative number(~-700) Maximal relative error is ~3e-7 for single-precision output
cvFastArctan	Fast arctangent calculation
cvCbrt	Fast cubic root calculation
cvCheckArr	Checks array values for NaNs, Infs or simply for too large numbers
cvRandArr	Random mat
cvRandShuffle	Random mat
cvSort	Sort mat for 1-D array
cvRSort	A rebol test
cvSolveCubic	Finds real roots of a cubic equation
cvSolvePoly	Finds all real and complex roots of a polynomial equation
cvCrossProduct	Calculates cross product of two 3d vectors
cvGEMM	Extended matrix transform: dst = alpha*op(A)*op(B) + beta*op(C), where op(X) is X or X^T
cvMatMulAdd	Inline function: Matrix transform: dst = A*B + C, C is optional
cvMatMul	idem
cvMatMulAddEx	Idem
cvTransform	Transforms each element of source array and stores resultant vectors in destination array
cvPerspectiveTransform	Does perspective transform on every element of input array
cvMulTransposed	Calculates (A-delta)*(A-delta)^T (order=0) or (A-delta)^T*(A-delta) (order=1)
cvTranspose	Tranposes matrix. Square matrices can be transposed in-place
cvMatMulAddS	Idem to cvTransform
cvFlip	Mirror array data around horizontal (flip=0), vertical (flip=1) or both(flip=-1) axises: cvFlip(src) flips images vertically and sequences horizontally (inplace)
cvMirror	cvFlip(src) flips images vertically and sequences horizontally (inplace)
cvSVD	Performs Singular Value Decomposition of a matrix
cvSVBkSb	Performs Singular Value Back Substitution (solves A*X = B): flags must be the same as in cvSVD
cvInvert	Inverts matrix (CV_32F OR CV_64F images!)

cvSolve	Solves linear system (src1)*(dst) = (src2) (returns 0 if src1 is a singular and CV_LU method is used)
cvDet	Calculates determinant of input matrix
cvTrace	Calculates trace of the matrix (sum of elements on the main diagonal). Return: cvScalar!
cvEigenVV	Finds eigen values and vectors of a symmetric matrix
cvSetIdentity	Makes an identity matrix (mat_ij = i == j)
cvRange	Fills matrix with given range of number
cvCalcCovarMatrix	Calculates covariation matrix for a set of vectors
cvCalcPCA	Performs Principal Component Analysis of a vector set
cvProjectPCA	Projects vectors to the specified subspace
cvBackProjectPCA	Reconstructs the original vectors from the projection coefficients
cvMahalanobis	Calculates Mahalanobis(weighted) distance
cvSum	Returns scalar: pbs
cvCountNonZero	Calculates number of non-zero pixels. The array, must be single-channel array or multi-channel image with COI set.
cvAvg	Calculates mean value of array elements Returns scalar
cvAvgSdv	Calculates mean and standard deviation of pixel values
cvMinMaxLoc	Finds global minimum, maximum and their positions
cvNorm	Finds norm, difference norm or relative difference norm for an array (or two arrays)
cvNormalize	idem
cvReduce	idem
cvDFT	Discrete Fourier Transform: complex->complex,real->ccs (forward),ccs->real (inverse)
cvFFT	Discrete Fourier Transform: complex->complex,real->ccs (forward),ccs->real (inverse)
cvMulSpectrums	Multiply results of DFTs: DFT(X)*DFT(Y) or DFT(X)*conj(DFT(Y))
cvGetOptimalDFTSize	Finds optimal DFT vector size >= size0
cvDCT	Discrete Cosine Transform
cvSliceLength	Calculates length of sequence slice (with support of negative indices)

cvCreateMemStorage	Creates new memory storage. block_size = 0 means that default,somewhat optimal size, is used (currently, it is 64K)
cvCreateChildMemStorage	Creates a memory storage that will borrow memory blocks from parent storage
cvReleaseMemStorage	Releases memory storage. All the children of a parent must be released before the parent. ;A child storage returns all the blocks to parent when it is released
cvClearMemStorage	Clears memory storage. This is the only way(!!!) (besides cvRestoreMemStoragePos) to reuse memory allocated for the storage - cvClearSeq,cvClearSet ... do not free any memory.A child storage returns all the blocks to the parent when it is cleared
cvSaveMemStoragePos	Remember a storage "free memory" position
cvRestoreMemStoragePos	Restore a storage "free memory" position
cvMemStorageAlloc	Allocates continuous buffer of the specified size in the storage
cvMemStorageAllocString	Allocates string in memory storage
cvCreateSeq	Creates new empty sequence that will reside in the specified storage
cvSetSeqBlockSize	Changes default size (granularity) of sequence blocks. The default size is ~1Kbyte
cvSeqPush	Adds new element to the end of sequence. Returns pointer to the element
cvSeqPushFront	Adds new element to the beginning of sequence. Returns pointer to it
cvSeqPop	Removes the last element from sequence and optionally saves it
cvSeqPopFront	Removes the first element from sequence and optioanally saves it
cvSeqPushMulti	Adds several new elements to the end of sequence
cvSeqPopMulti	;Removes several elements from the end of sequence and optionally saves them
cvSeqInsert	Inserts a new element in the middle of sequence.cvSeqInsert(seq,0,elem) == cvSeqPushFront(seq,elem)
cvSeqRemove	Removes specified sequence element
cvClearSeq	Removes all the elements from the sequence. The freed memory can be reused later only by the same sequence unless cvClearMemStorage or cvRestoreMemStoragePos is called}

cvGetSeqElem	Retrives pointer to specified sequence element. Negative indices are supported and mean counting from the end (e.g -1 means the last sequence element)
cvSeqElemIdx	Calculates index of the specified sequence element. Returns -1 if element does not belong to the sequence
cvStartAppendToSeq	Initializes sequence writer. The new elements will be added to the end of sequence
cvStartWriteSeq	Combination of cvCreateSeq and cvStartAppendToSeq
cvEndWriteSeq	loses sequence writer, updates sequence header and returns pointer to the resultant sequence (which may be useful if the sequence was created using cvStartWriteSeq))
cvFlushSeqWriter	Updates sequence header. May be useful to get access to some of previously written elements via cvGetSeqElem or sequence reader
cvStartReadSeq	Initializes sequence reader. The sequence can be read in forward or backward direction
cvGetSeqReaderPos	Returns current sequence reader position (currently observed sequence element)
cvSetSeqReaderPos	Changes sequence reader position. It may seek to an absolute or to relative to the current position
cvCvtSeqToArray	Copies sequence content to a continuous piece of memory
cvMakeSeqHeaderForArray	Creates sequence header for array. After that all the operations on sequences that do not alter the content can be applied to the resultant sequence
cvSeqSlice	Extracts sequence slice (with or without copying sequence elements)
cvCloneSeq	online
cvSeqRemoveSlice	Removes sequence slice
cvSeqInsertSlice	Inserts a sequence or array into another sequence
CvCmpFunc	a < b ? -1 : a > b ? 1 : 0
cvSeqSort	Sorts sequence in-place given element comparison function
cvSeqSearch	Finds element in a [sorted] sequence
cvSeqInvert	Reverses order of sequence elements in-place

cvSeqPartition	Splits sequence into one or more equivalence classes using the specified criteria
cvChangeSeqBlock	Internal sequence functions
cvCreateSeqBlock	Internal sequence functions
cvCreateSet	Creates a new set
cvSetAdd	Adds new element to the set and returns pointer to it
cvSetRemoveByPtr	inline Removes set element given its pointer
cvSetRemove	Removes element from the set by its index
cvGetSetElem	inline func Returns a set element by index. If the element doesn't belong to the set,NULL is returned
cvClearSet	Removes all the elements from the set
cvCreateGraph	Creates new graph
cvGraphAddVtx	Adds new vertex to the graph
cvGraphRemoveVtx	Removes vertex from the graph together with all incident edges
cvGraphRemoveVtxByPtr	idem
cvGraphAddEdge	Link two vertices specifed by indices or pointers
cvGraphAddEdgeByPtr	idem
cvGraphRemoveEdge	Remove edge connecting two vertices
cvGraphRemoveEdgeByPtr	Remove edge connecting two vertices
cvFindGraphEdge	Find edge connecting two vertices
cvFindGraphEdgeByPtr	idem
cvGraphFindEdge	alias
cvGraphFindEdgeByPtr	alias
cvClearGraph	Remove all vertices and edges from the graph
cvGraphVtxDegree	Count number of edges incident to the vertex
cvGraphVtxDegreeByPtr	idem
cvGetGraphVtx	Inline: Retrieves graph vertex by given index
cvGraphVtxIdx	Inline: Retrieves index of a graph vertex given its pointer
cvGraphEdgeIdxInline:	Retrieves index of a graph edge given its pointer
cvGraphGetVtxCount	idem
cvGraphGetEdgeCount	idem
cvCreateGraphScanner	Creates new graph scanner.
cvReleaseGraphScanner	Releases graph scanner
cvNextGraphItem	Get next graph element
cvCloneGraph	Creates a copy of graph
cvLUT	;Does look-up transformation. Elements of the source array (that should be 8uC1 or

	8sC1) are used as indexes in lutarr 256-element table
cvInitTreeNodeIterator	Iteration through the sequence tree
cvNextTreeNode	
cvPrevTreeNode	
cvInsertNodeIntoTree	Inserts sequence into tree
cvRemoveNodeFromTree	Removes contour from tree (together with the contour children).
cvTreeToNodeSeq	Gathers pointers to all the sequences
	The function implements the K-means algorithm for clustering an array of sample vectors in a specified number of classes
cvKMeans2	The function implements the K-means algorithm for clustering an array of sample vectors in a specified number of classes
cvUseOptimized	Loads optimized functions from IPP, MKL etc. or switches back to pure C code
cvOpenFileStorage	opens existing or creates new file storage
cvReleaseFileStorage	closes file storage and deallocates buffers
cvAttrValue	returns attribute value or 0 (NULL) if there is no such attribute
cvStartWriteStruct	starts writing compound structure (map or sequence)
cvEndWriteStruct	finishes writing compound structure
cvWriteInt	writes an integer
cvWriteReal	writes a floating-point number
cvWriteString	writes a string
cvWriteComment	writes a comment
cvWrite	writes instance of a standard type (matrix, image, sequence, graph etc.)
cvStartNextStream	starts the next stream
cvWriteRawData	helper function: writes multiple integer or floating-point numbers
cvGetHashedKey	returns the hash entry corresponding to the specified literal key string or 0 if there is no such a key in the storage
cvGetFileNode	returns file node with the specified key within the specified map
cvGetFileNodeByName	this is a slower version of cvGetFileNode that takes the key as a literal string
cvReadInt,cvReadIntByName, cvReadReal, cvReadRealByName, cvReadString,cvReadStringByName, cvReadByName	Not implemented inline functions.Red can be used for.
cvRead	decodes standard or user-defined object and returns it

cvStartReadRawData	starts reading data from sequence or scalar numeric node
cvReadRawDataSlice	reads multiple numbers and stores them to array
cvReadRawData	combination of two previous functions for easier reading of whole sequences
cvWriteFileNode	writes a copy of file node to file storage
cvGetFileNodeName	returns name of file node
cvRegisterType	Adding own types
cvUnregisterType	Adding own types
cvFirstType	Adding own types
cvFindType	Adding own types
cvTypeOf	Adding own types
cvRelease	universal functions
cvClone	universal functions
cvSave	simple API for reading/writing data
cvLoad	simple API for reading/writing data
cvGetTickCount	helper functions for RNG initialization and accurate time measurement: uses internal clock counter on x86
cvGetTickFrequency	idem
cvCheckHardwareSupport	CPU capabilities
cvGetNumThreads	retrieve/set the number of threads used in OpenMP implementations
cvSetNumThreads	idem
cvGetThreadNum	get index of the thread being executed
cvGetErrStatus	Get current OpenCV error status
cvSetErrStatus	Sets error status silently
cvGetErrMode	Retrives current error processing mode
cvSetErrMode	Sets error processing mode, returns previously used mode
cvError	Sets error status and performs some additonal actions (displaying message box, writing message to stderr, terminating application etc.)
cvErrorStr	Retrieves textual description of the error given its code
cvGetErrInfo	Retrieves detailed information about the last error occured
cvErrorFromIppStatus	Maps IPP error codes to the counterparts from OpenCV
cvRedirectError	Assigns a new error-handling function
cvNulDevReport	Output to nothing
cvStdErrReport	Output to console(fprintf(stderr,...))
cvGuiBoxReport	Output to MessageBox(WIN32)