JavaScript is disabled on your browser.

* [Overview](http://docs.google.com/overview-summary.html)
* [Package](http://docs.google.com/package-summary.html)
* Class
* [Tree](http://docs.google.com/package-tree.html)
* [Index](http://docs.google.com/index-all.html)
* [Help](http://docs.google.com/help-doc.html)
* [Prev Class](http://docs.google.com/org/opencv/video/SparsePyrLKOpticalFlow.html)
* Next Class
* [Frames](http://docs.google.com/index.html?org/opencv/video/Video.html)
* [No Frames](http://docs.google.com/Video.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* [Field](#3znysh7) |
* [Constr](#2et92p0) |
* [Method](#tyjcwt)
* Detail:
* [Field](#1t3h5sf) |
* [Constr](#1ksv4uv) |
* [Method](#2jxsxqh)

org.opencv.video

## Class Video

* java.lang.Object
  + org.opencv.video.Video
* public class Video  
  extends java.lang.Object

### Field SummaryFields

| Modifier and Type | Field and Description |
| --- | --- |
| static int | [**MOTION\_AFFINE**](http://docs.google.com/org/opencv/video/Video.html#MOTION_AFFINE) |
| static int | [**MOTION\_EUCLIDEAN**](http://docs.google.com/org/opencv/video/Video.html#MOTION_EUCLIDEAN) |
| static int | [**MOTION\_HOMOGRAPHY**](http://docs.google.com/org/opencv/video/Video.html#MOTION_HOMOGRAPHY) |
| static int | [**MOTION\_TRANSLATION**](http://docs.google.com/org/opencv/video/Video.html#MOTION_TRANSLATION) |
| static int | [**OPTFLOW\_FARNEBACK\_GAUSSIAN**](http://docs.google.com/org/opencv/video/Video.html#OPTFLOW_FARNEBACK_GAUSSIAN) |
| static int | [**OPTFLOW\_LK\_GET\_MIN\_EIGENVALS**](http://docs.google.com/org/opencv/video/Video.html#OPTFLOW_LK_GET_MIN_EIGENVALS) |
| static int | [**OPTFLOW\_USE\_INITIAL\_FLOW**](http://docs.google.com/org/opencv/video/Video.html#OPTFLOW_USE_INITIAL_FLOW) |

### Constructor SummaryConstructors

| Constructor and Description |
| --- |
| [**Video**](http://docs.google.com/org/opencv/video/Video.html#Video())() |

### Method SummaryMethods

| Modifier and Type | Method and Description |
| --- | --- |
| static int | [**buildOpticalFlowPyramid**](http://docs.google.com/org/opencv/video/Video.html#buildOpticalFlowPyramid(org.opencv.core.Mat,%20java.util.List,%20org.opencv.core.Size,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> pyramid, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int maxLevel) Constructs the image pyramid which can be passed to calcOpticalFlowPyrLK. |
| static int | [**buildOpticalFlowPyramid**](http://docs.google.com/org/opencv/video/Video.html#buildOpticalFlowPyramid(org.opencv.core.Mat,%20java.util.List,%20org.opencv.core.Size,%20int,%20boolean))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> pyramid, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int maxLevel, boolean withDerivatives) Constructs the image pyramid which can be passed to calcOpticalFlowPyrLK. |
| static int | [**buildOpticalFlowPyramid**](http://docs.google.com/org/opencv/video/Video.html#buildOpticalFlowPyramid(org.opencv.core.Mat,%20java.util.List,%20org.opencv.core.Size,%20int,%20boolean,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> pyramid, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int maxLevel, boolean withDerivatives, int pyrBorder) Constructs the image pyramid which can be passed to calcOpticalFlowPyrLK. |
| static int | [**buildOpticalFlowPyramid**](http://docs.google.com/org/opencv/video/Video.html#buildOpticalFlowPyramid(org.opencv.core.Mat,%20java.util.List,%20org.opencv.core.Size,%20int,%20boolean,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> pyramid, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int maxLevel, boolean withDerivatives, int pyrBorder, int derivBorder) Constructs the image pyramid which can be passed to calcOpticalFlowPyrLK. |
| static int | [**buildOpticalFlowPyramid**](http://docs.google.com/org/opencv/video/Video.html#buildOpticalFlowPyramid(org.opencv.core.Mat,%20java.util.List,%20org.opencv.core.Size,%20int,%20boolean,%20int,%20int,%20boolean))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> pyramid, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int maxLevel, boolean withDerivatives, int pyrBorder, int derivBorder, boolean tryReuseInputImage) Constructs the image pyramid which can be passed to calcOpticalFlowPyrLK. |
| static void | [**calcOpticalFlowFarneback**](http://docs.google.com/org/opencv/video/Video.html#calcOpticalFlowFarneback(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20int,%20int,%20int,%20int,%20double,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) prev, [Mat](http://docs.google.com/org/opencv/core/Mat.html) next, [Mat](http://docs.google.com/org/opencv/core/Mat.html) flow, double pyr\_scale, int levels, int winsize, int iterations, int poly\_n, double poly\_sigma, int flags) Computes a dense optical flow using the Gunnar Farneback's algorithm. |
| static void | [**calcOpticalFlowPyrLK**](http://docs.google.com/org/opencv/video/Video.html#calcOpticalFlowPyrLK(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.MatOfPoint2f,%20org.opencv.core.MatOfPoint2f,%20org.opencv.core.MatOfByte,%20org.opencv.core.MatOfFloat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) prevImg, [Mat](http://docs.google.com/org/opencv/core/Mat.html) nextImg, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) prevPts, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) nextPts, [MatOfByte](http://docs.google.com/org/opencv/core/MatOfByte.html) status, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) err) Calculates an optical flow for a sparse feature set using the iterative Lucas-Kanade method with pyramids. |
| static void | [**calcOpticalFlowPyrLK**](http://docs.google.com/org/opencv/video/Video.html#calcOpticalFlowPyrLK(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.MatOfPoint2f,%20org.opencv.core.MatOfPoint2f,%20org.opencv.core.MatOfByte,%20org.opencv.core.MatOfFloat,%20org.opencv.core.Size))([Mat](http://docs.google.com/org/opencv/core/Mat.html) prevImg, [Mat](http://docs.google.com/org/opencv/core/Mat.html) nextImg, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) prevPts, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) nextPts, [MatOfByte](http://docs.google.com/org/opencv/core/MatOfByte.html) status, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) err, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize) Calculates an optical flow for a sparse feature set using the iterative Lucas-Kanade method with pyramids. |
| static void | [**calcOpticalFlowPyrLK**](http://docs.google.com/org/opencv/video/Video.html#calcOpticalFlowPyrLK(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.MatOfPoint2f,%20org.opencv.core.MatOfPoint2f,%20org.opencv.core.MatOfByte,%20org.opencv.core.MatOfFloat,%20org.opencv.core.Size,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) prevImg, [Mat](http://docs.google.com/org/opencv/core/Mat.html) nextImg, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) prevPts, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) nextPts, [MatOfByte](http://docs.google.com/org/opencv/core/MatOfByte.html) status, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) err, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int maxLevel) Calculates an optical flow for a sparse feature set using the iterative Lucas-Kanade method with pyramids. |
| static void | [**calcOpticalFlowPyrLK**](http://docs.google.com/org/opencv/video/Video.html#calcOpticalFlowPyrLK(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.MatOfPoint2f,%20org.opencv.core.MatOfPoint2f,%20org.opencv.core.MatOfByte,%20org.opencv.core.MatOfFloat,%20org.opencv.core.Size,%20int,%20org.opencv.core.TermCriteria))([Mat](http://docs.google.com/org/opencv/core/Mat.html) prevImg, [Mat](http://docs.google.com/org/opencv/core/Mat.html) nextImg, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) prevPts, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) nextPts, [MatOfByte](http://docs.google.com/org/opencv/core/MatOfByte.html) status, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) err, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int maxLevel, [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) criteria) Calculates an optical flow for a sparse feature set using the iterative Lucas-Kanade method with pyramids. |
| static void | [**calcOpticalFlowPyrLK**](http://docs.google.com/org/opencv/video/Video.html#calcOpticalFlowPyrLK(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.MatOfPoint2f,%20org.opencv.core.MatOfPoint2f,%20org.opencv.core.MatOfByte,%20org.opencv.core.MatOfFloat,%20org.opencv.core.Size,%20int,%20org.opencv.core.TermCriteria,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) prevImg, [Mat](http://docs.google.com/org/opencv/core/Mat.html) nextImg, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) prevPts, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) nextPts, [MatOfByte](http://docs.google.com/org/opencv/core/MatOfByte.html) status, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) err, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int maxLevel, [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) criteria, int flags) Calculates an optical flow for a sparse feature set using the iterative Lucas-Kanade method with pyramids. |
| static void | [**calcOpticalFlowPyrLK**](http://docs.google.com/org/opencv/video/Video.html#calcOpticalFlowPyrLK(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.MatOfPoint2f,%20org.opencv.core.MatOfPoint2f,%20org.opencv.core.MatOfByte,%20org.opencv.core.MatOfFloat,%20org.opencv.core.Size,%20int,%20org.opencv.core.TermCriteria,%20int,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) prevImg, [Mat](http://docs.google.com/org/opencv/core/Mat.html) nextImg, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) prevPts, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) nextPts, [MatOfByte](http://docs.google.com/org/opencv/core/MatOfByte.html) status, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) err, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int maxLevel, [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) criteria, int flags, double minEigThreshold) Calculates an optical flow for a sparse feature set using the iterative Lucas-Kanade method with pyramids. |
| static [RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) | [**CamShift**](http://docs.google.com/org/opencv/video/Video.html#CamShift(org.opencv.core.Mat,%20org.opencv.core.Rect,%20org.opencv.core.TermCriteria))([Mat](http://docs.google.com/org/opencv/core/Mat.html) probImage, [Rect](http://docs.google.com/org/opencv/core/Rect.html) window, [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) criteria) Finds an object center, size, and orientation. |
| static double | [**computeECC**](http://docs.google.com/org/opencv/video/Video.html#computeECC(org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) templateImage, [Mat](http://docs.google.com/org/opencv/core/Mat.html) inputImage) Computes the Enhanced Correlation Coefficient value between two images CITE: EP08 . |
| static double | [**computeECC**](http://docs.google.com/org/opencv/video/Video.html#computeECC(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) templateImage, [Mat](http://docs.google.com/org/opencv/core/Mat.html) inputImage, [Mat](http://docs.google.com/org/opencv/core/Mat.html) inputMask) Computes the Enhanced Correlation Coefficient value between two images CITE: EP08 . |
| static [BackgroundSubtractorKNN](http://docs.google.com/org/opencv/video/BackgroundSubtractorKNN.html) | [**createBackgroundSubtractorKNN**](http://docs.google.com/org/opencv/video/Video.html#createBackgroundSubtractorKNN())() Creates KNN Background Subtractor whether a pixel is close to that sample. |
| static [BackgroundSubtractorKNN](http://docs.google.com/org/opencv/video/BackgroundSubtractorKNN.html) | [**createBackgroundSubtractorKNN**](http://docs.google.com/org/opencv/video/Video.html#createBackgroundSubtractorKNN(int))(int history) Creates KNN Background Subtractor |
| static [BackgroundSubtractorKNN](http://docs.google.com/org/opencv/video/BackgroundSubtractorKNN.html) | [**createBackgroundSubtractorKNN**](http://docs.google.com/org/opencv/video/Video.html#createBackgroundSubtractorKNN(int,%20double))(int history, double dist2Threshold) Creates KNN Background Subtractor |
| static [BackgroundSubtractorKNN](http://docs.google.com/org/opencv/video/BackgroundSubtractorKNN.html) | [**createBackgroundSubtractorKNN**](http://docs.google.com/org/opencv/video/Video.html#createBackgroundSubtractorKNN(int,%20double,%20boolean))(int history, double dist2Threshold, boolean detectShadows) Creates KNN Background Subtractor |
| static [BackgroundSubtractorMOG2](http://docs.google.com/org/opencv/video/BackgroundSubtractorMOG2.html) | [**createBackgroundSubtractorMOG2**](http://docs.google.com/org/opencv/video/Video.html#createBackgroundSubtractorMOG2())() Creates MOG2 Background Subtractor to decide whether a pixel is well described by the background model. |
| static [BackgroundSubtractorMOG2](http://docs.google.com/org/opencv/video/BackgroundSubtractorMOG2.html) | [**createBackgroundSubtractorMOG2**](http://docs.google.com/org/opencv/video/Video.html#createBackgroundSubtractorMOG2(int))(int history) Creates MOG2 Background Subtractor |
| static [BackgroundSubtractorMOG2](http://docs.google.com/org/opencv/video/BackgroundSubtractorMOG2.html) | [**createBackgroundSubtractorMOG2**](http://docs.google.com/org/opencv/video/Video.html#createBackgroundSubtractorMOG2(int,%20double))(int history, double varThreshold) Creates MOG2 Background Subtractor |
| static [BackgroundSubtractorMOG2](http://docs.google.com/org/opencv/video/BackgroundSubtractorMOG2.html) | [**createBackgroundSubtractorMOG2**](http://docs.google.com/org/opencv/video/Video.html#createBackgroundSubtractorMOG2(int,%20double,%20boolean))(int history, double varThreshold, boolean detectShadows) Creates MOG2 Background Subtractor |
| static [DualTVL1OpticalFlow](http://docs.google.com/org/opencv/video/DualTVL1OpticalFlow.html) | [**createOptFlow\_DualTVL1**](http://docs.google.com/org/opencv/video/Video.html#createOptFlow_DualTVL1())() Creates instance of cv::DenseOpticalFlow |
| static [Mat](http://docs.google.com/org/opencv/core/Mat.html) | [**estimateRigidTransform**](http://docs.google.com/org/opencv/video/Video.html#estimateRigidTransform(org.opencv.core.Mat,%20org.opencv.core.Mat,%20boolean))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, boolean fullAffine) Computes an optimal affine transformation between two 2D point sets. |
| static [Mat](http://docs.google.com/org/opencv/core/Mat.html) | [**estimateRigidTransform**](http://docs.google.com/org/opencv/video/Video.html#estimateRigidTransform(org.opencv.core.Mat,%20org.opencv.core.Mat,%20boolean,%20int,%20double,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, boolean fullAffine, int ransacMaxIters, double ransacGoodRatio, int ransacSize0) |
| static double | [**findTransformECC**](http://docs.google.com/org/opencv/video/Video.html#findTransformECC(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.TermCriteria,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) templateImage, [Mat](http://docs.google.com/org/opencv/core/Mat.html) inputImage, [Mat](http://docs.google.com/org/opencv/core/Mat.html) warpMatrix, int motionType, [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) criteria, [Mat](http://docs.google.com/org/opencv/core/Mat.html) inputMask, int gaussFiltSize) Finds the geometric transform (warp) between two images in terms of the ECC criterion CITE: EP08 . |
| static int | [**meanShift**](http://docs.google.com/org/opencv/video/Video.html#meanShift(org.opencv.core.Mat,%20org.opencv.core.Rect,%20org.opencv.core.TermCriteria))([Mat](http://docs.google.com/org/opencv/core/Mat.html) probImage, [Rect](http://docs.google.com/org/opencv/core/Rect.html) window, [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) criteria) Finds an object on a back projection image. |

### Methods inherited from class java.lang.Objectequals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Field Detail

#### MOTION\_AFFINE public static final int MOTION\_AFFINESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.video.Video.MOTION_AFFINE)

#### MOTION\_EUCLIDEAN public static final int MOTION\_EUCLIDEANSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.video.Video.MOTION_EUCLIDEAN)

#### MOTION\_HOMOGRAPHY public static final int MOTION\_HOMOGRAPHYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.video.Video.MOTION_HOMOGRAPHY)

#### MOTION\_TRANSLATION public static final int MOTION\_TRANSLATIONSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.video.Video.MOTION_TRANSLATION)

#### OPTFLOW\_FARNEBACK\_GAUSSIAN public static final int OPTFLOW\_FARNEBACK\_GAUSSIANSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.video.Video.OPTFLOW_FARNEBACK_GAUSSIAN)

#### OPTFLOW\_LK\_GET\_MIN\_EIGENVALS public static final int OPTFLOW\_LK\_GET\_MIN\_EIGENVALSSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.video.Video.OPTFLOW_LK_GET_MIN_EIGENVALS)

#### OPTFLOW\_USE\_INITIAL\_FLOW public static final int OPTFLOW\_USE\_INITIAL\_FLOWSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.video.Video.OPTFLOW_USE_INITIAL_FLOW)

### Constructor Detail

#### Video public Video()

### Method Detail

#### buildOpticalFlowPyramid public static int buildOpticalFlowPyramid([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> pyramid, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int maxLevel) Constructs the image pyramid which can be passed to calcOpticalFlowPyrLK.Parameters:img - 8-bit input image.pyramid - output pyramid.winSize - window size of optical flow algorithm. Must be not less than winSize argument of calcOpticalFlowPyrLK. It is needed to calculate required padding for pyramid levels.maxLevel - 0-based maximal pyramid level number. constructed without the gradients then calcOpticalFlowPyrLK will calculate them internally. to force data copying. Returns:number of levels in constructed pyramid. Can be less than maxLevel.

#### buildOpticalFlowPyramid public static int buildOpticalFlowPyramid([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> pyramid, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int maxLevel, boolean withDerivatives) Constructs the image pyramid which can be passed to calcOpticalFlowPyrLK.Parameters:img - 8-bit input image.pyramid - output pyramid.winSize - window size of optical flow algorithm. Must be not less than winSize argument of calcOpticalFlowPyrLK. It is needed to calculate required padding for pyramid levels.maxLevel - 0-based maximal pyramid level number.withDerivatives - set to precompute gradients for the every pyramid level. If pyramid is constructed without the gradients then calcOpticalFlowPyrLK will calculate them internally. to force data copying. Returns:number of levels in constructed pyramid. Can be less than maxLevel.

#### buildOpticalFlowPyramid public static int buildOpticalFlowPyramid([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> pyramid, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int maxLevel, boolean withDerivatives, int pyrBorder) Constructs the image pyramid which can be passed to calcOpticalFlowPyrLK.Parameters:img - 8-bit input image.pyramid - output pyramid.winSize - window size of optical flow algorithm. Must be not less than winSize argument of calcOpticalFlowPyrLK. It is needed to calculate required padding for pyramid levels.maxLevel - 0-based maximal pyramid level number.withDerivatives - set to precompute gradients for the every pyramid level. If pyramid is constructed without the gradients then calcOpticalFlowPyrLK will calculate them internally.pyrBorder - the border mode for pyramid layers. to force data copying. Returns:number of levels in constructed pyramid. Can be less than maxLevel.

#### buildOpticalFlowPyramid public static int buildOpticalFlowPyramid([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> pyramid, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int maxLevel, boolean withDerivatives, int pyrBorder, int derivBorder) Constructs the image pyramid which can be passed to calcOpticalFlowPyrLK.Parameters:img - 8-bit input image.pyramid - output pyramid.winSize - window size of optical flow algorithm. Must be not less than winSize argument of calcOpticalFlowPyrLK. It is needed to calculate required padding for pyramid levels.maxLevel - 0-based maximal pyramid level number.withDerivatives - set to precompute gradients for the every pyramid level. If pyramid is constructed without the gradients then calcOpticalFlowPyrLK will calculate them internally.pyrBorder - the border mode for pyramid layers.derivBorder - the border mode for gradients. to force data copying. Returns:number of levels in constructed pyramid. Can be less than maxLevel.

#### buildOpticalFlowPyramid public static int buildOpticalFlowPyramid([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> pyramid, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int maxLevel, boolean withDerivatives, int pyrBorder, int derivBorder, boolean tryReuseInputImage) Constructs the image pyramid which can be passed to calcOpticalFlowPyrLK.Parameters:img - 8-bit input image.pyramid - output pyramid.winSize - window size of optical flow algorithm. Must be not less than winSize argument of calcOpticalFlowPyrLK. It is needed to calculate required padding for pyramid levels.maxLevel - 0-based maximal pyramid level number.withDerivatives - set to precompute gradients for the every pyramid level. If pyramid is constructed without the gradients then calcOpticalFlowPyrLK will calculate them internally.pyrBorder - the border mode for pyramid layers.derivBorder - the border mode for gradients.tryReuseInputImage - put ROI of input image into the pyramid if possible. You can pass false to force data copying. Returns:number of levels in constructed pyramid. Can be less than maxLevel.

#### calcOpticalFlowFarneback public static void calcOpticalFlowFarneback([Mat](http://docs.google.com/org/opencv/core/Mat.html) prev, [Mat](http://docs.google.com/org/opencv/core/Mat.html) next, [Mat](http://docs.google.com/org/opencv/core/Mat.html) flow, double pyr\_scale, int levels, int winsize, int iterations, int poly\_n, double poly\_sigma, int flags) Computes a dense optical flow using the Gunnar Farneback's algorithm.Parameters:prev - first 8-bit single-channel input image.next - second input image of the same size and the same type as prev.flow - computed flow image that has the same size as prev and type CV\_32FC2.pyr\_scale - parameter, specifying the image scale (<1) to build pyramids for each image; pyr\_scale=0.5 means a classical pyramid, where each next layer is twice smaller than the previous one.levels - number of pyramid layers including the initial image; levels=1 means that no extra layers are created and only the original images are used.winsize - averaging window size; larger values increase the algorithm robustness to image noise and give more chances for fast motion detection, but yield more blurred motion field.iterations - number of iterations the algorithm does at each pyramid level.poly\_n - size of the pixel neighborhood used to find polynomial expansion in each pixel; larger values mean that the image will be approximated with smoother surfaces, yielding more robust algorithm and more blurred motion field, typically poly\_n =5 or 7.poly\_sigma - standard deviation of the Gaussian that is used to smooth derivatives used as a basis for the polynomial expansion; for poly\_n=5, you can set poly\_sigma=1.1, for poly\_n=7, a good value would be poly\_sigma=1.5.flags - operation flags that can be a combination of the following:

* + - * **OPTFLOW\_USE\_INITIAL\_FLOW** uses the input flow as an initial flow approximation.
      * **OPTFLOW\_FARNEBACK\_GAUSSIAN** uses the Gaussian \(\texttt{winsize}\times\texttt{winsize}\) filter instead of a box filter of the same size for optical flow estimation; usually, this option gives z more accurate flow than with a box filter, at the cost of lower speed; normally, winsize for a Gaussian window should be set to a larger value to achieve the same level of robustness.

The function finds an optical flow for each prev pixel using the CITE: Farneback2003 algorithm so that \(\texttt{prev} (y,x) \sim \texttt{next} ( y + \texttt{flow} (y,x)[1], x + \texttt{flow} (y,x)[0])\) **Note:**

* + - * An example using the optical flow algorithm described by Gunnar Farneback can be found at opencv\_source\_code/samples/cpp/fback.cpp
      * (Python) An example using the optical flow algorithm described by Gunnar Farneback can be found at opencv\_source\_code/samples/python/opt\_flow.py

#### calcOpticalFlowPyrLK public static void calcOpticalFlowPyrLK([Mat](http://docs.google.com/org/opencv/core/Mat.html) prevImg, [Mat](http://docs.google.com/org/opencv/core/Mat.html) nextImg, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) prevPts, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) nextPts, [MatOfByte](http://docs.google.com/org/opencv/core/MatOfByte.html) status, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) err) Calculates an optical flow for a sparse feature set using the iterative Lucas-Kanade method with pyramids.Parameters:prevImg - first 8-bit input image or pyramid constructed by buildOpticalFlowPyramid.nextImg - second input image or pyramid of the same size and the same type as prevImg.prevPts - vector of 2D points for which the flow needs to be found; point coordinates must be single-precision floating-point numbers.nextPts - output vector of 2D points (with single-precision floating-point coordinates) containing the calculated new positions of input features in the second image; when OPTFLOW\_USE\_INITIAL\_FLOW flag is passed, the vector must have the same size as in the input.status - output status vector (of unsigned chars); each element of the vector is set to 1 if the flow for the corresponding features has been found, otherwise, it is set to 0.err - output vector of errors; each element of the vector is set to an error for the corresponding feature, type of the error measure can be set in flags parameter; if the flow wasn't found then the error is not defined (use the status parameter to find such cases). level), if set to 1, two levels are used, and so on; if pyramids are passed to input then algorithm will use as many levels as pyramids have but no more than maxLevel. (after the specified maximum number of iterations criteria.maxCount or when the search window moves by less than criteria.epsilon.

* + - * **OPTFLOW\_USE\_INITIAL\_FLOW** uses initial estimations, stored in nextPts; if the flag is not set, then prevPts is copied to nextPts and is considered the initial estimate.
      * **OPTFLOW\_LK\_GET\_MIN\_EIGENVALS** use minimum eigen values as an error measure (see minEigThreshold description); if the flag is not set, then L1 distance between patches around the original and a moved point, divided by number of pixels in a window, is used as a error measure. optical flow equations (this matrix is called a spatial gradient matrix in CITE: Bouguet00), divided by number of pixels in a window; if this value is less than minEigThreshold, then a corresponding feature is filtered out and its flow is not processed, so it allows to remove bad points and get a performance boost.

The function implements a sparse iterative version of the Lucas-Kanade optical flow in pyramids. See CITE: Bouguet00 . The function is parallelized with the TBB library. **Note:**

* + - * An example using the Lucas-Kanade optical flow algorithm can be found at opencv\_source\_code/samples/cpp/lkdemo.cpp
      * (Python) An example using the Lucas-Kanade optical flow algorithm can be found at opencv\_source\_code/samples/python/lk\_track.py
      * (Python) An example using the Lucas-Kanade tracker for homography matching can be found at opencv\_source\_code/samples/python/lk\_homography.py

#### calcOpticalFlowPyrLK public static void calcOpticalFlowPyrLK([Mat](http://docs.google.com/org/opencv/core/Mat.html) prevImg, [Mat](http://docs.google.com/org/opencv/core/Mat.html) nextImg, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) prevPts, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) nextPts, [MatOfByte](http://docs.google.com/org/opencv/core/MatOfByte.html) status, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) err, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize) Calculates an optical flow for a sparse feature set using the iterative Lucas-Kanade method with pyramids.Parameters:prevImg - first 8-bit input image or pyramid constructed by buildOpticalFlowPyramid.nextImg - second input image or pyramid of the same size and the same type as prevImg.prevPts - vector of 2D points for which the flow needs to be found; point coordinates must be single-precision floating-point numbers.nextPts - output vector of 2D points (with single-precision floating-point coordinates) containing the calculated new positions of input features in the second image; when OPTFLOW\_USE\_INITIAL\_FLOW flag is passed, the vector must have the same size as in the input.status - output status vector (of unsigned chars); each element of the vector is set to 1 if the flow for the corresponding features has been found, otherwise, it is set to 0.err - output vector of errors; each element of the vector is set to an error for the corresponding feature, type of the error measure can be set in flags parameter; if the flow wasn't found then the error is not defined (use the status parameter to find such cases).winSize - size of the search window at each pyramid level. level), if set to 1, two levels are used, and so on; if pyramids are passed to input then algorithm will use as many levels as pyramids have but no more than maxLevel. (after the specified maximum number of iterations criteria.maxCount or when the search window moves by less than criteria.epsilon.

* + - * **OPTFLOW\_USE\_INITIAL\_FLOW** uses initial estimations, stored in nextPts; if the flag is not set, then prevPts is copied to nextPts and is considered the initial estimate.
      * **OPTFLOW\_LK\_GET\_MIN\_EIGENVALS** use minimum eigen values as an error measure (see minEigThreshold description); if the flag is not set, then L1 distance between patches around the original and a moved point, divided by number of pixels in a window, is used as a error measure. optical flow equations (this matrix is called a spatial gradient matrix in CITE: Bouguet00), divided by number of pixels in a window; if this value is less than minEigThreshold, then a corresponding feature is filtered out and its flow is not processed, so it allows to remove bad points and get a performance boost.

The function implements a sparse iterative version of the Lucas-Kanade optical flow in pyramids. See CITE: Bouguet00 . The function is parallelized with the TBB library. **Note:**

* + - * An example using the Lucas-Kanade optical flow algorithm can be found at opencv\_source\_code/samples/cpp/lkdemo.cpp
      * (Python) An example using the Lucas-Kanade optical flow algorithm can be found at opencv\_source\_code/samples/python/lk\_track.py
      * (Python) An example using the Lucas-Kanade tracker for homography matching can be found at opencv\_source\_code/samples/python/lk\_homography.py

#### calcOpticalFlowPyrLK public static void calcOpticalFlowPyrLK([Mat](http://docs.google.com/org/opencv/core/Mat.html) prevImg, [Mat](http://docs.google.com/org/opencv/core/Mat.html) nextImg, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) prevPts, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) nextPts, [MatOfByte](http://docs.google.com/org/opencv/core/MatOfByte.html) status, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) err, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int maxLevel) Calculates an optical flow for a sparse feature set using the iterative Lucas-Kanade method with pyramids.Parameters:prevImg - first 8-bit input image or pyramid constructed by buildOpticalFlowPyramid.nextImg - second input image or pyramid of the same size and the same type as prevImg.prevPts - vector of 2D points for which the flow needs to be found; point coordinates must be single-precision floating-point numbers.nextPts - output vector of 2D points (with single-precision floating-point coordinates) containing the calculated new positions of input features in the second image; when OPTFLOW\_USE\_INITIAL\_FLOW flag is passed, the vector must have the same size as in the input.status - output status vector (of unsigned chars); each element of the vector is set to 1 if the flow for the corresponding features has been found, otherwise, it is set to 0.err - output vector of errors; each element of the vector is set to an error for the corresponding feature, type of the error measure can be set in flags parameter; if the flow wasn't found then the error is not defined (use the status parameter to find such cases).winSize - size of the search window at each pyramid level.maxLevel - 0-based maximal pyramid level number; if set to 0, pyramids are not used (single level), if set to 1, two levels are used, and so on; if pyramids are passed to input then algorithm will use as many levels as pyramids have but no more than maxLevel. (after the specified maximum number of iterations criteria.maxCount or when the search window moves by less than criteria.epsilon.

* + - * **OPTFLOW\_USE\_INITIAL\_FLOW** uses initial estimations, stored in nextPts; if the flag is not set, then prevPts is copied to nextPts and is considered the initial estimate.
      * **OPTFLOW\_LK\_GET\_MIN\_EIGENVALS** use minimum eigen values as an error measure (see minEigThreshold description); if the flag is not set, then L1 distance between patches around the original and a moved point, divided by number of pixels in a window, is used as a error measure. optical flow equations (this matrix is called a spatial gradient matrix in CITE: Bouguet00), divided by number of pixels in a window; if this value is less than minEigThreshold, then a corresponding feature is filtered out and its flow is not processed, so it allows to remove bad points and get a performance boost.

The function implements a sparse iterative version of the Lucas-Kanade optical flow in pyramids. See CITE: Bouguet00 . The function is parallelized with the TBB library. **Note:**

* + - * An example using the Lucas-Kanade optical flow algorithm can be found at opencv\_source\_code/samples/cpp/lkdemo.cpp
      * (Python) An example using the Lucas-Kanade optical flow algorithm can be found at opencv\_source\_code/samples/python/lk\_track.py
      * (Python) An example using the Lucas-Kanade tracker for homography matching can be found at opencv\_source\_code/samples/python/lk\_homography.py

#### calcOpticalFlowPyrLK public static void calcOpticalFlowPyrLK([Mat](http://docs.google.com/org/opencv/core/Mat.html) prevImg, [Mat](http://docs.google.com/org/opencv/core/Mat.html) nextImg, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) prevPts, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) nextPts, [MatOfByte](http://docs.google.com/org/opencv/core/MatOfByte.html) status, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) err, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int maxLevel, [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) criteria) Calculates an optical flow for a sparse feature set using the iterative Lucas-Kanade method with pyramids.Parameters:prevImg - first 8-bit input image or pyramid constructed by buildOpticalFlowPyramid.nextImg - second input image or pyramid of the same size and the same type as prevImg.prevPts - vector of 2D points for which the flow needs to be found; point coordinates must be single-precision floating-point numbers.nextPts - output vector of 2D points (with single-precision floating-point coordinates) containing the calculated new positions of input features in the second image; when OPTFLOW\_USE\_INITIAL\_FLOW flag is passed, the vector must have the same size as in the input.status - output status vector (of unsigned chars); each element of the vector is set to 1 if the flow for the corresponding features has been found, otherwise, it is set to 0.err - output vector of errors; each element of the vector is set to an error for the corresponding feature, type of the error measure can be set in flags parameter; if the flow wasn't found then the error is not defined (use the status parameter to find such cases).winSize - size of the search window at each pyramid level.maxLevel - 0-based maximal pyramid level number; if set to 0, pyramids are not used (single level), if set to 1, two levels are used, and so on; if pyramids are passed to input then algorithm will use as many levels as pyramids have but no more than maxLevel.criteria - parameter, specifying the termination criteria of the iterative search algorithm (after the specified maximum number of iterations criteria.maxCount or when the search window moves by less than criteria.epsilon.

* + - * **OPTFLOW\_USE\_INITIAL\_FLOW** uses initial estimations, stored in nextPts; if the flag is not set, then prevPts is copied to nextPts and is considered the initial estimate.
      * **OPTFLOW\_LK\_GET\_MIN\_EIGENVALS** use minimum eigen values as an error measure (see minEigThreshold description); if the flag is not set, then L1 distance between patches around the original and a moved point, divided by number of pixels in a window, is used as a error measure. optical flow equations (this matrix is called a spatial gradient matrix in CITE: Bouguet00), divided by number of pixels in a window; if this value is less than minEigThreshold, then a corresponding feature is filtered out and its flow is not processed, so it allows to remove bad points and get a performance boost.

The function implements a sparse iterative version of the Lucas-Kanade optical flow in pyramids. See CITE: Bouguet00 . The function is parallelized with the TBB library. **Note:**

* + - * An example using the Lucas-Kanade optical flow algorithm can be found at opencv\_source\_code/samples/cpp/lkdemo.cpp
      * (Python) An example using the Lucas-Kanade optical flow algorithm can be found at opencv\_source\_code/samples/python/lk\_track.py
      * (Python) An example using the Lucas-Kanade tracker for homography matching can be found at opencv\_source\_code/samples/python/lk\_homography.py

#### calcOpticalFlowPyrLK public static void calcOpticalFlowPyrLK([Mat](http://docs.google.com/org/opencv/core/Mat.html) prevImg, [Mat](http://docs.google.com/org/opencv/core/Mat.html) nextImg, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) prevPts, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) nextPts, [MatOfByte](http://docs.google.com/org/opencv/core/MatOfByte.html) status, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) err, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int maxLevel, [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) criteria, int flags) Calculates an optical flow for a sparse feature set using the iterative Lucas-Kanade method with pyramids.Parameters:prevImg - first 8-bit input image or pyramid constructed by buildOpticalFlowPyramid.nextImg - second input image or pyramid of the same size and the same type as prevImg.prevPts - vector of 2D points for which the flow needs to be found; point coordinates must be single-precision floating-point numbers.nextPts - output vector of 2D points (with single-precision floating-point coordinates) containing the calculated new positions of input features in the second image; when OPTFLOW\_USE\_INITIAL\_FLOW flag is passed, the vector must have the same size as in the input.status - output status vector (of unsigned chars); each element of the vector is set to 1 if the flow for the corresponding features has been found, otherwise, it is set to 0.err - output vector of errors; each element of the vector is set to an error for the corresponding feature, type of the error measure can be set in flags parameter; if the flow wasn't found then the error is not defined (use the status parameter to find such cases).winSize - size of the search window at each pyramid level.maxLevel - 0-based maximal pyramid level number; if set to 0, pyramids are not used (single level), if set to 1, two levels are used, and so on; if pyramids are passed to input then algorithm will use as many levels as pyramids have but no more than maxLevel.criteria - parameter, specifying the termination criteria of the iterative search algorithm (after the specified maximum number of iterations criteria.maxCount or when the search window moves by less than criteria.epsilon.flags - operation flags:

* + - * **OPTFLOW\_USE\_INITIAL\_FLOW** uses initial estimations, stored in nextPts; if the flag is not set, then prevPts is copied to nextPts and is considered the initial estimate.
      * **OPTFLOW\_LK\_GET\_MIN\_EIGENVALS** use minimum eigen values as an error measure (see minEigThreshold description); if the flag is not set, then L1 distance between patches around the original and a moved point, divided by number of pixels in a window, is used as a error measure. optical flow equations (this matrix is called a spatial gradient matrix in CITE: Bouguet00), divided by number of pixels in a window; if this value is less than minEigThreshold, then a corresponding feature is filtered out and its flow is not processed, so it allows to remove bad points and get a performance boost.

The function implements a sparse iterative version of the Lucas-Kanade optical flow in pyramids. See CITE: Bouguet00 . The function is parallelized with the TBB library. **Note:**

* + - * An example using the Lucas-Kanade optical flow algorithm can be found at opencv\_source\_code/samples/cpp/lkdemo.cpp
      * (Python) An example using the Lucas-Kanade optical flow algorithm can be found at opencv\_source\_code/samples/python/lk\_track.py
      * (Python) An example using the Lucas-Kanade tracker for homography matching can be found at opencv\_source\_code/samples/python/lk\_homography.py

#### calcOpticalFlowPyrLK public static void calcOpticalFlowPyrLK([Mat](http://docs.google.com/org/opencv/core/Mat.html) prevImg, [Mat](http://docs.google.com/org/opencv/core/Mat.html) nextImg, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) prevPts, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) nextPts, [MatOfByte](http://docs.google.com/org/opencv/core/MatOfByte.html) status, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) err, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int maxLevel, [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) criteria, int flags, double minEigThreshold) Calculates an optical flow for a sparse feature set using the iterative Lucas-Kanade method with pyramids.Parameters:prevImg - first 8-bit input image or pyramid constructed by buildOpticalFlowPyramid.nextImg - second input image or pyramid of the same size and the same type as prevImg.prevPts - vector of 2D points for which the flow needs to be found; point coordinates must be single-precision floating-point numbers.nextPts - output vector of 2D points (with single-precision floating-point coordinates) containing the calculated new positions of input features in the second image; when OPTFLOW\_USE\_INITIAL\_FLOW flag is passed, the vector must have the same size as in the input.status - output status vector (of unsigned chars); each element of the vector is set to 1 if the flow for the corresponding features has been found, otherwise, it is set to 0.err - output vector of errors; each element of the vector is set to an error for the corresponding feature, type of the error measure can be set in flags parameter; if the flow wasn't found then the error is not defined (use the status parameter to find such cases).winSize - size of the search window at each pyramid level.maxLevel - 0-based maximal pyramid level number; if set to 0, pyramids are not used (single level), if set to 1, two levels are used, and so on; if pyramids are passed to input then algorithm will use as many levels as pyramids have but no more than maxLevel.criteria - parameter, specifying the termination criteria of the iterative search algorithm (after the specified maximum number of iterations criteria.maxCount or when the search window moves by less than criteria.epsilon.flags - operation flags:

* + - * **OPTFLOW\_USE\_INITIAL\_FLOW** uses initial estimations, stored in nextPts; if the flag is not set, then prevPts is copied to nextPts and is considered the initial estimate.
      * **OPTFLOW\_LK\_GET\_MIN\_EIGENVALS** use minimum eigen values as an error measure (see minEigThreshold description); if the flag is not set, then L1 distance between patches around the original and a moved point, divided by number of pixels in a window, is used as a error measure.

minEigThreshold - the algorithm calculates the minimum eigen value of a 2x2 normal matrix of optical flow equations (this matrix is called a spatial gradient matrix in CITE: Bouguet00), divided by number of pixels in a window; if this value is less than minEigThreshold, then a corresponding feature is filtered out and its flow is not processed, so it allows to remove bad points and get a performance boost.The function implements a sparse iterative version of the Lucas-Kanade optical flow in pyramids. See CITE: Bouguet00 . The function is parallelized with the TBB library. **Note:**

* + - An example using the Lucas-Kanade optical flow algorithm can be found at opencv\_source\_code/samples/cpp/lkdemo.cpp
    - (Python) An example using the Lucas-Kanade optical flow algorithm can be found at opencv\_source\_code/samples/python/lk\_track.py
    - (Python) An example using the Lucas-Kanade tracker for homography matching can be found at opencv\_source\_code/samples/python/lk\_homography.py

#### CamShift public static [RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) CamShift([Mat](http://docs.google.com/org/opencv/core/Mat.html) probImage, [Rect](http://docs.google.com/org/opencv/core/Rect.html) window, [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) criteria) Finds an object center, size, and orientation.Parameters:probImage - Back projection of the object histogram. See calcBackProject.window - Initial search window.criteria - Stop criteria for the underlying meanShift. returns (in old interfaces) Number of iterations CAMSHIFT took to converge The function implements the CAMSHIFT object tracking algorithm CITE: Bradski98 . First, it finds an object center using meanShift and then adjusts the window size and finds the optimal rotation. The function returns the rotated rectangle structure that includes the object position, size, and orientation. The next position of the search window can be obtained with RotatedRect::boundingRect() See the OpenCV sample camshiftdemo.c that tracks colored objects. **Note:**

* + - (Python) A sample explaining the camshift tracking algorithm can be found at opencv\_source\_code/samples/python/camshift.py

Returns:automatically generated

#### computeECC public static double computeECC([Mat](http://docs.google.com/org/opencv/core/Mat.html) templateImage, [Mat](http://docs.google.com/org/opencv/core/Mat.html) inputImage) Computes the Enhanced Correlation Coefficient value between two images CITE: EP08 .Parameters:templateImage - single-channel template image; CV\_8U or CV\_32F array.inputImage - single-channel input image to be warped to provide an image similar to templateImage, same type as templateImage. SEE: findTransformECC Returns:automatically generated

#### computeECC public static double computeECC([Mat](http://docs.google.com/org/opencv/core/Mat.html) templateImage, [Mat](http://docs.google.com/org/opencv/core/Mat.html) inputImage, [Mat](http://docs.google.com/org/opencv/core/Mat.html) inputMask) Computes the Enhanced Correlation Coefficient value between two images CITE: EP08 .Parameters:templateImage - single-channel template image; CV\_8U or CV\_32F array.inputImage - single-channel input image to be warped to provide an image similar to templateImage, same type as templateImage.inputMask - An optional mask to indicate valid values of inputImage. SEE: findTransformECC Returns:automatically generated

#### createBackgroundSubtractorKNN public static [BackgroundSubtractorKNN](http://docs.google.com/org/opencv/video/BackgroundSubtractorKNN.html) createBackgroundSubtractorKNN() Creates KNN Background Subtractor whether a pixel is close to that sample. This parameter does not affect the background update. speed a bit, so if you do not need this feature, set the parameter to false.Returns:automatically generated

#### createBackgroundSubtractorKNN public static [BackgroundSubtractorKNN](http://docs.google.com/org/opencv/video/BackgroundSubtractorKNN.html) createBackgroundSubtractorKNN(int history) Creates KNN Background SubtractorParameters:history - Length of the history. whether a pixel is close to that sample. This parameter does not affect the background update. speed a bit, so if you do not need this feature, set the parameter to false. Returns:automatically generated

#### createBackgroundSubtractorKNN public static [BackgroundSubtractorKNN](http://docs.google.com/org/opencv/video/BackgroundSubtractorKNN.html) createBackgroundSubtractorKNN(int history, double dist2Threshold) Creates KNN Background SubtractorParameters:history - Length of the history.dist2Threshold - Threshold on the squared distance between the pixel and the sample to decide whether a pixel is close to that sample. This parameter does not affect the background update. speed a bit, so if you do not need this feature, set the parameter to false. Returns:automatically generated

#### createBackgroundSubtractorKNN public static [BackgroundSubtractorKNN](http://docs.google.com/org/opencv/video/BackgroundSubtractorKNN.html) createBackgroundSubtractorKNN(int history, double dist2Threshold, boolean detectShadows) Creates KNN Background SubtractorParameters:history - Length of the history.dist2Threshold - Threshold on the squared distance between the pixel and the sample to decide whether a pixel is close to that sample. This parameter does not affect the background update.detectShadows - If true, the algorithm will detect shadows and mark them. It decreases the speed a bit, so if you do not need this feature, set the parameter to false. Returns:automatically generated

#### createBackgroundSubtractorMOG2 public static [BackgroundSubtractorMOG2](http://docs.google.com/org/opencv/video/BackgroundSubtractorMOG2.html) createBackgroundSubtractorMOG2() Creates MOG2 Background Subtractor to decide whether a pixel is well described by the background model. This parameter does not affect the background update. speed a bit, so if you do not need this feature, set the parameter to false.Returns:automatically generated

#### createBackgroundSubtractorMOG2 public static [BackgroundSubtractorMOG2](http://docs.google.com/org/opencv/video/BackgroundSubtractorMOG2.html) createBackgroundSubtractorMOG2(int history) Creates MOG2 Background SubtractorParameters:history - Length of the history. to decide whether a pixel is well described by the background model. This parameter does not affect the background update. speed a bit, so if you do not need this feature, set the parameter to false. Returns:automatically generated

#### createBackgroundSubtractorMOG2 public static [BackgroundSubtractorMOG2](http://docs.google.com/org/opencv/video/BackgroundSubtractorMOG2.html) createBackgroundSubtractorMOG2(int history, double varThreshold) Creates MOG2 Background SubtractorParameters:history - Length of the history.varThreshold - Threshold on the squared Mahalanobis distance between the pixel and the model to decide whether a pixel is well described by the background model. This parameter does not affect the background update. speed a bit, so if you do not need this feature, set the parameter to false. Returns:automatically generated

#### createBackgroundSubtractorMOG2 public static [BackgroundSubtractorMOG2](http://docs.google.com/org/opencv/video/BackgroundSubtractorMOG2.html) createBackgroundSubtractorMOG2(int history, double varThreshold, boolean detectShadows) Creates MOG2 Background SubtractorParameters:history - Length of the history.varThreshold - Threshold on the squared Mahalanobis distance between the pixel and the model to decide whether a pixel is well described by the background model. This parameter does not affect the background update.detectShadows - If true, the algorithm will detect shadows and mark them. It decreases the speed a bit, so if you do not need this feature, set the parameter to false. Returns:automatically generated

#### createOptFlow\_DualTVL1 public static [DualTVL1OpticalFlow](http://docs.google.com/org/opencv/video/DualTVL1OpticalFlow.html) createOptFlow\_DualTVL1() Creates instance of cv::DenseOpticalFlowReturns:automatically generated

#### estimateRigidTransform public static [Mat](http://docs.google.com/org/opencv/core/Mat.html) estimateRigidTransform([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, boolean fullAffine) Computes an optimal affine transformation between two 2D point sets.Parameters:src - First input 2D point set stored in std::vector or Mat, or an image stored in Mat.dst - Second input 2D point set of the same size and the same type as A, or another image.fullAffine - If true, the function finds an optimal affine transformation with no additional restrictions (6 degrees of freedom). Otherwise, the class of transformations to choose from is limited to combinations of translation, rotation, and uniform scaling (4 degrees of freedom). The function finds an optimal affine transform \*[A|b]\* (a 2 x 3 floating-point matrix) that approximates best the affine transformation between: Two point sets Two raster images. In this case, the function first finds some features in the src image and finds the corresponding features in dst image. After that, the problem is reduced to the first case. In case of point sets, the problem is formulated as follows: you need to find a 2x2 matrix \*A\* and 2x1 vector \*b\* so that: \([A^\*|b^\*] = arg \min \_{[A|b]} \sum \_i \| \texttt{dst}[i] - A { \texttt{src}[i]}^T - b \| ^2\) where src[i] and dst[i] are the i-th points in src and dst, respectively \([A|b]\) can be either arbitrary (when fullAffine=true ) or have a form of \(\begin{bmatrix} a\_{11} & a\_{12} & b\_1 \\ -a\_{12} & a\_{11} & b\_2 \end{bmatrix}\) when fullAffine=false. SEE: estimateAffine2D, estimateAffinePartial2D, getAffineTransform, getPerspectiveTransform, findHomography Returns:automatically generated

#### estimateRigidTransform public static [Mat](http://docs.google.com/org/opencv/core/Mat.html) estimateRigidTransform([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, boolean fullAffine, int ransacMaxIters, double ransacGoodRatio, int ransacSize0)

#### findTransformECC public static double findTransformECC([Mat](http://docs.google.com/org/opencv/core/Mat.html) templateImage, [Mat](http://docs.google.com/org/opencv/core/Mat.html) inputImage, [Mat](http://docs.google.com/org/opencv/core/Mat.html) warpMatrix, int motionType, [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) criteria, [Mat](http://docs.google.com/org/opencv/core/Mat.html) inputMask, int gaussFiltSize) Finds the geometric transform (warp) between two images in terms of the ECC criterion CITE: EP08 .Parameters:templateImage - single-channel template image; CV\_8U or CV\_32F array.inputImage - single-channel input image which should be warped with the final warpMatrix in order to provide an image similar to templateImage, same type as templateImage.warpMatrix - floating-point \(2\times 3\) or \(3\times 3\) mapping matrix (warp).motionType - parameter, specifying the type of motion:

* + - **MOTION\_TRANSLATION** sets a translational motion model; warpMatrix is \(2\times 3\) with the first \(2\times 2\) part being the unity matrix and the rest two parameters being estimated.
    - **MOTION\_EUCLIDEAN** sets a Euclidean (rigid) transformation as motion model; three parameters are estimated; warpMatrix is \(2\times 3\).
    - **MOTION\_AFFINE** sets an affine motion model (DEFAULT); six parameters are estimated; warpMatrix is \(2\times 3\).
    - **MOTION\_HOMOGRAPHY** sets a homography as a motion model; eight parameters are estimated;\warpMatrix\ is \(3\times 3\).

criteria - parameter, specifying the termination criteria of the ECC algorithm; criteria.epsilon defines the threshold of the increment in the correlation coefficient between two iterations (a negative criteria.epsilon makes criteria.maxcount the only termination criterion). Default values are shown in the declaration above.inputMask - An optional mask to indicate valid values of inputImage.gaussFiltSize - An optional value indicating size of gaussian blur filter; (DEFAULT: 5)The function estimates the optimum transformation (warpMatrix) with respect to ECC criterion (CITE: EP08), that is \(\texttt{warpMatrix} = \arg\max\_{W} \texttt{ECC}(\texttt{templateImage}(x,y),\texttt{inputImage}(x',y'))\) where \(\begin{bmatrix} x' \\ y' \end{bmatrix} = W \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}\) (the equation holds with homogeneous coordinates for homography). It returns the final enhanced correlation coefficient, that is the correlation coefficient between the template image and the final warped input image. When a \(3\times 3\) matrix is given with motionType =0, 1 or 2, the third row is ignored. Unlike findHomography and estimateRigidTransform, the function findTransformECC implements an area-based alignment that builds on intensity similarities. In essence, the function updates the initial transformation that roughly aligns the images. If this information is missing, the identity warp (unity matrix) is used as an initialization. Note that if images undergo strong displacements/rotations, an initial transformation that roughly aligns the images is necessary (e.g., a simple euclidean/similarity transform that allows for the images showing the same image content approximately). Use inverse warping in the second image to take an image close to the first one, i.e. use the flag WARP\_INVERSE\_MAP with warpAffine or warpPerspective. See also the OpenCV sample image\_alignment.cpp that demonstrates the use of the function. Note that the function throws an exception if algorithm does not converges. SEE: computeECC, estimateAffine2D, estimateAffinePartial2D, findHomography Returns:automatically generated

#### meanShift public static int meanShift([Mat](http://docs.google.com/org/opencv/core/Mat.html) probImage, [Rect](http://docs.google.com/org/opencv/core/Rect.html) window, [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) criteria) Finds an object on a back projection image.Parameters:probImage - Back projection of the object histogram. See calcBackProject for details.window - Initial search window.criteria - Stop criteria for the iterative search algorithm. returns : Number of iterations CAMSHIFT took to converge. The function implements the iterative object search algorithm. It takes the input back projection of an object and the initial position. The mass center in window of the back projection image is computed and the search window center shifts to the mass center. The procedure is repeated until the specified number of iterations criteria.maxCount is done or until the window center shifts by less than criteria.epsilon. The algorithm is used inside CamShift and, unlike CamShift , the search window size or orientation do not change during the search. You can simply pass the output of calcBackProject to this function. But better results can be obtained if you pre-filter the back projection and remove the noise. For example, you can do this by retrieving connected components with findContours , throwing away contours with small area ( contourArea ), and rendering the remaining contours with drawContours. Returns:automatically generated

* [Overview](http://docs.google.com/overview-summary.html)
* [Package](http://docs.google.com/package-summary.html)
* Class
* [Tree](http://docs.google.com/package-tree.html)
* [Index](http://docs.google.com/index-all.html)
* [Help](http://docs.google.com/help-doc.html)
* [Prev Class](http://docs.google.com/org/opencv/video/SparsePyrLKOpticalFlow.html)
* Next Class
* [Frames](http://docs.google.com/index.html?org/opencv/video/Video.html)
* [No Frames](http://docs.google.com/Video.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* [Field](#3znysh7) |
* [Constr](#2et92p0) |
* [Method](#tyjcwt)
* Detail:
* [Field](#1t3h5sf) |
* [Constr](#1ksv4uv) |
* [Method](#2jxsxqh)

Generated on 2021-04-02 03:15:03 / OpenCV 3.4.14