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/imgproc/GeneralizedHoughGuil.html)
* [Next Class](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html)
* [Frames](http://docs.google.com/index.html?org/opencv/imgproc/Imgproc.html)
* [No Frames](http://docs.google.com/Imgproc.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* [Field](#3znysh7) |
* [Constr](#2et92p0) |
* [Method](#tyjcwt)
* Detail:
* [Field](#1t3h5sf) |
* [Constr](#1pr1rn1) |
* [Method](#2ovzkin)

org.opencv.imgproc

## Class Imgproc

* java.lang.Object
  + org.opencv.imgproc.Imgproc
* public class Imgproc  
  extends java.lang.Object

### Field SummaryFields

| Modifier and Type | Field and Description |
| --- | --- |
| static int | [**ADAPTIVE\_THRESH\_GAUSSIAN\_C**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#ADAPTIVE_THRESH_GAUSSIAN_C) |
| static int | [**ADAPTIVE\_THRESH\_MEAN\_C**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#ADAPTIVE_THRESH_MEAN_C) |
| static int | [**CC\_STAT\_AREA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CC_STAT_AREA) |
| static int | [**CC\_STAT\_HEIGHT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CC_STAT_HEIGHT) |
| static int | [**CC\_STAT\_LEFT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CC_STAT_LEFT) |
| static int | [**CC\_STAT\_MAX**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CC_STAT_MAX) |
| static int | [**CC\_STAT\_TOP**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CC_STAT_TOP) |
| static int | [**CC\_STAT\_WIDTH**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CC_STAT_WIDTH) |
| static int | [**CCL\_BBDT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CCL_BBDT) |
| static int | [**CCL\_BOLELLI**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CCL_BOLELLI) |
| static int | [**CCL\_DEFAULT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CCL_DEFAULT) |
| static int | [**CCL\_GRANA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CCL_GRANA) |
| static int | [**CCL\_SAUF**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CCL_SAUF) |
| static int | [**CCL\_SPAGHETTI**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CCL_SPAGHETTI) |
| static int | [**CCL\_WU**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CCL_WU) |
| static int | [**CHAIN\_APPROX\_NONE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CHAIN_APPROX_NONE) |
| static int | [**CHAIN\_APPROX\_SIMPLE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CHAIN_APPROX_SIMPLE) |
| static int | [**CHAIN\_APPROX\_TC89\_KCOS**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CHAIN_APPROX_TC89_KCOS) |
| static int | [**CHAIN\_APPROX\_TC89\_L1**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CHAIN_APPROX_TC89_L1) |
| static int | [**COLOR\_BayerBG2BGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerBG2BGR) |
| static int | [**COLOR\_BayerBG2BGR\_EA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerBG2BGR_EA) |
| static int | [**COLOR\_BayerBG2BGR\_VNG**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerBG2BGR_VNG) |
| static int | [**COLOR\_BayerBG2BGRA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerBG2BGRA) |
| static int | [**COLOR\_BayerBG2GRAY**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerBG2GRAY) |
| static int | [**COLOR\_BayerBG2RGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerBG2RGB) |
| static int | [**COLOR\_BayerBG2RGB\_EA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerBG2RGB_EA) |
| static int | [**COLOR\_BayerBG2RGB\_VNG**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerBG2RGB_VNG) |
| static int | [**COLOR\_BayerBG2RGBA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerBG2RGBA) |
| static int | [**COLOR\_BayerGB2BGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerGB2BGR) |
| static int | [**COLOR\_BayerGB2BGR\_EA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerGB2BGR_EA) |
| static int | [**COLOR\_BayerGB2BGR\_VNG**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerGB2BGR_VNG) |
| static int | [**COLOR\_BayerGB2BGRA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerGB2BGRA) |
| static int | [**COLOR\_BayerGB2GRAY**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerGB2GRAY) |
| static int | [**COLOR\_BayerGB2RGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerGB2RGB) |
| static int | [**COLOR\_BayerGB2RGB\_EA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerGB2RGB_EA) |
| static int | [**COLOR\_BayerGB2RGB\_VNG**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerGB2RGB_VNG) |
| static int | [**COLOR\_BayerGB2RGBA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerGB2RGBA) |
| static int | [**COLOR\_BayerGR2BGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerGR2BGR) |
| static int | [**COLOR\_BayerGR2BGR\_EA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerGR2BGR_EA) |
| static int | [**COLOR\_BayerGR2BGR\_VNG**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerGR2BGR_VNG) |
| static int | [**COLOR\_BayerGR2BGRA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerGR2BGRA) |
| static int | [**COLOR\_BayerGR2GRAY**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerGR2GRAY) |
| static int | [**COLOR\_BayerGR2RGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerGR2RGB) |
| static int | [**COLOR\_BayerGR2RGB\_EA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerGR2RGB_EA) |
| static int | [**COLOR\_BayerGR2RGB\_VNG**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerGR2RGB_VNG) |
| static int | [**COLOR\_BayerGR2RGBA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerGR2RGBA) |
| static int | [**COLOR\_BayerRG2BGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerRG2BGR) |
| static int | [**COLOR\_BayerRG2BGR\_EA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerRG2BGR_EA) |
| static int | [**COLOR\_BayerRG2BGR\_VNG**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerRG2BGR_VNG) |
| static int | [**COLOR\_BayerRG2BGRA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerRG2BGRA) |
| static int | [**COLOR\_BayerRG2GRAY**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerRG2GRAY) |
| static int | [**COLOR\_BayerRG2RGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerRG2RGB) |
| static int | [**COLOR\_BayerRG2RGB\_EA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerRG2RGB_EA) |
| static int | [**COLOR\_BayerRG2RGB\_VNG**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerRG2RGB_VNG) |
| static int | [**COLOR\_BayerRG2RGBA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BayerRG2RGBA) |
| static int | [**COLOR\_BGR2BGR555**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR2BGR555) |
| static int | [**COLOR\_BGR2BGR565**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR2BGR565) |
| static int | [**COLOR\_BGR2BGRA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR2BGRA) |
| static int | [**COLOR\_BGR2GRAY**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR2GRAY) |
| static int | [**COLOR\_BGR2HLS**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR2HLS) |
| static int | [**COLOR\_BGR2HLS\_FULL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR2HLS_FULL) |
| static int | [**COLOR\_BGR2HSV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR2HSV) |
| static int | [**COLOR\_BGR2HSV\_FULL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR2HSV_FULL) |
| static int | [**COLOR\_BGR2Lab**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR2Lab) |
| static int | [**COLOR\_BGR2Luv**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR2Luv) |
| static int | [**COLOR\_BGR2RGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR2RGB) |
| static int | [**COLOR\_BGR2RGBA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR2RGBA) |
| static int | [**COLOR\_BGR2XYZ**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR2XYZ) |
| static int | [**COLOR\_BGR2YCrCb**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR2YCrCb) |
| static int | [**COLOR\_BGR2YUV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR2YUV) |
| static int | [**COLOR\_BGR2YUV\_I420**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR2YUV_I420) |
| static int | [**COLOR\_BGR2YUV\_IYUV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR2YUV_IYUV) |
| static int | [**COLOR\_BGR2YUV\_YV12**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR2YUV_YV12) |
| static int | [**COLOR\_BGR5552BGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR5552BGR) |
| static int | [**COLOR\_BGR5552BGRA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR5552BGRA) |
| static int | [**COLOR\_BGR5552GRAY**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR5552GRAY) |
| static int | [**COLOR\_BGR5552RGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR5552RGB) |
| static int | [**COLOR\_BGR5552RGBA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR5552RGBA) |
| static int | [**COLOR\_BGR5652BGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR5652BGR) |
| static int | [**COLOR\_BGR5652BGRA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR5652BGRA) |
| static int | [**COLOR\_BGR5652GRAY**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR5652GRAY) |
| static int | [**COLOR\_BGR5652RGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR5652RGB) |
| static int | [**COLOR\_BGR5652RGBA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGR5652RGBA) |
| static int | [**COLOR\_BGRA2BGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGRA2BGR) |
| static int | [**COLOR\_BGRA2BGR555**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGRA2BGR555) |
| static int | [**COLOR\_BGRA2BGR565**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGRA2BGR565) |
| static int | [**COLOR\_BGRA2GRAY**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGRA2GRAY) |
| static int | [**COLOR\_BGRA2RGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGRA2RGB) |
| static int | [**COLOR\_BGRA2RGBA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGRA2RGBA) |
| static int | [**COLOR\_BGRA2YUV\_I420**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGRA2YUV_I420) |
| static int | [**COLOR\_BGRA2YUV\_IYUV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGRA2YUV_IYUV) |
| static int | [**COLOR\_BGRA2YUV\_YV12**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_BGRA2YUV_YV12) |
| static int | [**COLOR\_COLORCVT\_MAX**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_COLORCVT_MAX) |
| static int | [**COLOR\_GRAY2BGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_GRAY2BGR) |
| static int | [**COLOR\_GRAY2BGR555**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_GRAY2BGR555) |
| static int | [**COLOR\_GRAY2BGR565**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_GRAY2BGR565) |
| static int | [**COLOR\_GRAY2BGRA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_GRAY2BGRA) |
| static int | [**COLOR\_GRAY2RGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_GRAY2RGB) |
| static int | [**COLOR\_GRAY2RGBA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_GRAY2RGBA) |
| static int | [**COLOR\_HLS2BGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_HLS2BGR) |
| static int | [**COLOR\_HLS2BGR\_FULL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_HLS2BGR_FULL) |
| static int | [**COLOR\_HLS2RGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_HLS2RGB) |
| static int | [**COLOR\_HLS2RGB\_FULL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_HLS2RGB_FULL) |
| static int | [**COLOR\_HSV2BGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_HSV2BGR) |
| static int | [**COLOR\_HSV2BGR\_FULL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_HSV2BGR_FULL) |
| static int | [**COLOR\_HSV2RGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_HSV2RGB) |
| static int | [**COLOR\_HSV2RGB\_FULL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_HSV2RGB_FULL) |
| static int | [**COLOR\_Lab2BGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_Lab2BGR) |
| static int | [**COLOR\_Lab2LBGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_Lab2LBGR) |
| static int | [**COLOR\_Lab2LRGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_Lab2LRGB) |
| static int | [**COLOR\_Lab2RGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_Lab2RGB) |
| static int | [**COLOR\_LBGR2Lab**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_LBGR2Lab) |
| static int | [**COLOR\_LBGR2Luv**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_LBGR2Luv) |
| static int | [**COLOR\_LRGB2Lab**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_LRGB2Lab) |
| static int | [**COLOR\_LRGB2Luv**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_LRGB2Luv) |
| static int | [**COLOR\_Luv2BGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_Luv2BGR) |
| static int | [**COLOR\_Luv2LBGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_Luv2LBGR) |
| static int | [**COLOR\_Luv2LRGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_Luv2LRGB) |
| static int | [**COLOR\_Luv2RGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_Luv2RGB) |
| static int | [**COLOR\_mRGBA2RGBA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_mRGBA2RGBA) |
| static int | [**COLOR\_RGB2BGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGB2BGR) |
| static int | [**COLOR\_RGB2BGR555**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGB2BGR555) |
| static int | [**COLOR\_RGB2BGR565**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGB2BGR565) |
| static int | [**COLOR\_RGB2BGRA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGB2BGRA) |
| static int | [**COLOR\_RGB2GRAY**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGB2GRAY) |
| static int | [**COLOR\_RGB2HLS**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGB2HLS) |
| static int | [**COLOR\_RGB2HLS\_FULL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGB2HLS_FULL) |
| static int | [**COLOR\_RGB2HSV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGB2HSV) |
| static int | [**COLOR\_RGB2HSV\_FULL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGB2HSV_FULL) |
| static int | [**COLOR\_RGB2Lab**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGB2Lab) |
| static int | [**COLOR\_RGB2Luv**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGB2Luv) |
| static int | [**COLOR\_RGB2RGBA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGB2RGBA) |
| static int | [**COLOR\_RGB2XYZ**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGB2XYZ) |
| static int | [**COLOR\_RGB2YCrCb**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGB2YCrCb) |
| static int | [**COLOR\_RGB2YUV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGB2YUV) |
| static int | [**COLOR\_RGB2YUV\_I420**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGB2YUV_I420) |
| static int | [**COLOR\_RGB2YUV\_IYUV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGB2YUV_IYUV) |
| static int | [**COLOR\_RGB2YUV\_YV12**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGB2YUV_YV12) |
| static int | [**COLOR\_RGBA2BGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGBA2BGR) |
| static int | [**COLOR\_RGBA2BGR555**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGBA2BGR555) |
| static int | [**COLOR\_RGBA2BGR565**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGBA2BGR565) |
| static int | [**COLOR\_RGBA2BGRA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGBA2BGRA) |
| static int | [**COLOR\_RGBA2GRAY**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGBA2GRAY) |
| static int | [**COLOR\_RGBA2mRGBA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGBA2mRGBA) |
| static int | [**COLOR\_RGBA2RGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGBA2RGB) |
| static int | [**COLOR\_RGBA2YUV\_I420**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGBA2YUV_I420) |
| static int | [**COLOR\_RGBA2YUV\_IYUV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGBA2YUV_IYUV) |
| static int | [**COLOR\_RGBA2YUV\_YV12**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_RGBA2YUV_YV12) |
| static int | [**COLOR\_XYZ2BGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_XYZ2BGR) |
| static int | [**COLOR\_XYZ2RGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_XYZ2RGB) |
| static int | [**COLOR\_YCrCb2BGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YCrCb2BGR) |
| static int | [**COLOR\_YCrCb2RGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YCrCb2RGB) |
| static int | [**COLOR\_YUV2BGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGR) |
| static int | [**COLOR\_YUV2BGR\_I420**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGR_I420) |
| static int | [**COLOR\_YUV2BGR\_IYUV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGR_IYUV) |
| static int | [**COLOR\_YUV2BGR\_NV12**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGR_NV12) |
| static int | [**COLOR\_YUV2BGR\_NV21**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGR_NV21) |
| static int | [**COLOR\_YUV2BGR\_UYNV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGR_UYNV) |
| static int | [**COLOR\_YUV2BGR\_UYVY**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGR_UYVY) |
| static int | [**COLOR\_YUV2BGR\_Y422**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGR_Y422) |
| static int | [**COLOR\_YUV2BGR\_YUNV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGR_YUNV) |
| static int | [**COLOR\_YUV2BGR\_YUY2**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGR_YUY2) |
| static int | [**COLOR\_YUV2BGR\_YUYV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGR_YUYV) |
| static int | [**COLOR\_YUV2BGR\_YV12**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGR_YV12) |
| static int | [**COLOR\_YUV2BGR\_YVYU**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGR_YVYU) |
| static int | [**COLOR\_YUV2BGRA\_I420**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGRA_I420) |
| static int | [**COLOR\_YUV2BGRA\_IYUV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGRA_IYUV) |
| static int | [**COLOR\_YUV2BGRA\_NV12**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGRA_NV12) |
| static int | [**COLOR\_YUV2BGRA\_NV21**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGRA_NV21) |
| static int | [**COLOR\_YUV2BGRA\_UYNV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGRA_UYNV) |
| static int | [**COLOR\_YUV2BGRA\_UYVY**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGRA_UYVY) |
| static int | [**COLOR\_YUV2BGRA\_Y422**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGRA_Y422) |
| static int | [**COLOR\_YUV2BGRA\_YUNV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGRA_YUNV) |
| static int | [**COLOR\_YUV2BGRA\_YUY2**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGRA_YUY2) |
| static int | [**COLOR\_YUV2BGRA\_YUYV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGRA_YUYV) |
| static int | [**COLOR\_YUV2BGRA\_YV12**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGRA_YV12) |
| static int | [**COLOR\_YUV2BGRA\_YVYU**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2BGRA_YVYU) |
| static int | [**COLOR\_YUV2GRAY\_420**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2GRAY_420) |
| static int | [**COLOR\_YUV2GRAY\_I420**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2GRAY_I420) |
| static int | [**COLOR\_YUV2GRAY\_IYUV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2GRAY_IYUV) |
| static int | [**COLOR\_YUV2GRAY\_NV12**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2GRAY_NV12) |
| static int | [**COLOR\_YUV2GRAY\_NV21**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2GRAY_NV21) |
| static int | [**COLOR\_YUV2GRAY\_UYNV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2GRAY_UYNV) |
| static int | [**COLOR\_YUV2GRAY\_UYVY**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2GRAY_UYVY) |
| static int | [**COLOR\_YUV2GRAY\_Y422**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2GRAY_Y422) |
| static int | [**COLOR\_YUV2GRAY\_YUNV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2GRAY_YUNV) |
| static int | [**COLOR\_YUV2GRAY\_YUY2**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2GRAY_YUY2) |
| static int | [**COLOR\_YUV2GRAY\_YUYV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2GRAY_YUYV) |
| static int | [**COLOR\_YUV2GRAY\_YV12**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2GRAY_YV12) |
| static int | [**COLOR\_YUV2GRAY\_YVYU**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2GRAY_YVYU) |
| static int | [**COLOR\_YUV2RGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGB) |
| static int | [**COLOR\_YUV2RGB\_I420**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGB_I420) |
| static int | [**COLOR\_YUV2RGB\_IYUV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGB_IYUV) |
| static int | [**COLOR\_YUV2RGB\_NV12**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGB_NV12) |
| static int | [**COLOR\_YUV2RGB\_NV21**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGB_NV21) |
| static int | [**COLOR\_YUV2RGB\_UYNV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGB_UYNV) |
| static int | [**COLOR\_YUV2RGB\_UYVY**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGB_UYVY) |
| static int | [**COLOR\_YUV2RGB\_Y422**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGB_Y422) |
| static int | [**COLOR\_YUV2RGB\_YUNV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGB_YUNV) |
| static int | [**COLOR\_YUV2RGB\_YUY2**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGB_YUY2) |
| static int | [**COLOR\_YUV2RGB\_YUYV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGB_YUYV) |
| static int | [**COLOR\_YUV2RGB\_YV12**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGB_YV12) |
| static int | [**COLOR\_YUV2RGB\_YVYU**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGB_YVYU) |
| static int | [**COLOR\_YUV2RGBA\_I420**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGBA_I420) |
| static int | [**COLOR\_YUV2RGBA\_IYUV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGBA_IYUV) |
| static int | [**COLOR\_YUV2RGBA\_NV12**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGBA_NV12) |
| static int | [**COLOR\_YUV2RGBA\_NV21**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGBA_NV21) |
| static int | [**COLOR\_YUV2RGBA\_UYNV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGBA_UYNV) |
| static int | [**COLOR\_YUV2RGBA\_UYVY**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGBA_UYVY) |
| static int | [**COLOR\_YUV2RGBA\_Y422**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGBA_Y422) |
| static int | [**COLOR\_YUV2RGBA\_YUNV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGBA_YUNV) |
| static int | [**COLOR\_YUV2RGBA\_YUY2**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGBA_YUY2) |
| static int | [**COLOR\_YUV2RGBA\_YUYV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGBA_YUYV) |
| static int | [**COLOR\_YUV2RGBA\_YV12**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGBA_YV12) |
| static int | [**COLOR\_YUV2RGBA\_YVYU**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV2RGBA_YVYU) |
| static int | [**COLOR\_YUV420p2BGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV420p2BGR) |
| static int | [**COLOR\_YUV420p2BGRA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV420p2BGRA) |
| static int | [**COLOR\_YUV420p2GRAY**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV420p2GRAY) |
| static int | [**COLOR\_YUV420p2RGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV420p2RGB) |
| static int | [**COLOR\_YUV420p2RGBA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV420p2RGBA) |
| static int | [**COLOR\_YUV420sp2BGR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV420sp2BGR) |
| static int | [**COLOR\_YUV420sp2BGRA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV420sp2BGRA) |
| static int | [**COLOR\_YUV420sp2GRAY**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV420sp2GRAY) |
| static int | [**COLOR\_YUV420sp2RGB**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV420sp2RGB) |
| static int | [**COLOR\_YUV420sp2RGBA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLOR_YUV420sp2RGBA) |
| static int | [**COLORMAP\_AUTUMN**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_AUTUMN) |
| static int | [**COLORMAP\_BONE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_BONE) |
| static int | [**COLORMAP\_CIVIDIS**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_CIVIDIS) |
| static int | [**COLORMAP\_COOL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_COOL) |
| static int | [**COLORMAP\_DEEPGREEN**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_DEEPGREEN) |
| static int | [**COLORMAP\_HOT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_HOT) |
| static int | [**COLORMAP\_HSV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_HSV) |
| static int | [**COLORMAP\_INFERNO**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_INFERNO) |
| static int | [**COLORMAP\_JET**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_JET) |
| static int | [**COLORMAP\_MAGMA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_MAGMA) |
| static int | [**COLORMAP\_OCEAN**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_OCEAN) |
| static int | [**COLORMAP\_PARULA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_PARULA) |
| static int | [**COLORMAP\_PINK**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_PINK) |
| static int | [**COLORMAP\_PLASMA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_PLASMA) |
| static int | [**COLORMAP\_RAINBOW**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_RAINBOW) |
| static int | [**COLORMAP\_SPRING**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_SPRING) |
| static int | [**COLORMAP\_SUMMER**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_SUMMER) |
| static int | [**COLORMAP\_TURBO**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_TURBO) |
| static int | [**COLORMAP\_TWILIGHT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_TWILIGHT) |
| static int | [**COLORMAP\_TWILIGHT\_SHIFTED**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_TWILIGHT_SHIFTED) |
| static int | [**COLORMAP\_VIRIDIS**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_VIRIDIS) |
| static int | [**COLORMAP\_WINTER**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#COLORMAP_WINTER) |
| static int | [**CONTOURS\_MATCH\_I1**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CONTOURS_MATCH_I1) |
| static int | [**CONTOURS\_MATCH\_I2**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CONTOURS_MATCH_I2) |
| static int | [**CONTOURS\_MATCH\_I3**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CONTOURS_MATCH_I3) |
| static int | [**CV\_BILATERAL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_BILATERAL) |
| static int | [**CV\_BLUR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_BLUR) |
| static int | [**CV\_BLUR\_NO\_SCALE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_BLUR_NO_SCALE) |
| static int | [**CV\_CANNY\_L2\_GRADIENT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_CANNY_L2_GRADIENT) |
| static int | [**CV\_CHAIN\_CODE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_CHAIN_CODE) |
| static int | [**CV\_CLOCKWISE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_CLOCKWISE) |
| static int | [**CV\_COMP\_BHATTACHARYYA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_COMP_BHATTACHARYYA) |
| static int | [**CV\_COMP\_CHISQR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_COMP_CHISQR) |
| static int | [**CV\_COMP\_CHISQR\_ALT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_COMP_CHISQR_ALT) |
| static int | [**CV\_COMP\_CORREL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_COMP_CORREL) |
| static int | [**CV\_COMP\_HELLINGER**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_COMP_HELLINGER) |
| static int | [**CV\_COMP\_INTERSECT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_COMP_INTERSECT) |
| static int | [**CV\_COMP\_KL\_DIV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_COMP_KL_DIV) |
| static int | [**CV\_CONTOURS\_MATCH\_I1**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_CONTOURS_MATCH_I1) |
| static int | [**CV\_CONTOURS\_MATCH\_I2**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_CONTOURS_MATCH_I2) |
| static int | [**CV\_CONTOURS\_MATCH\_I3**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_CONTOURS_MATCH_I3) |
| static int | [**CV\_COUNTER\_CLOCKWISE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_COUNTER_CLOCKWISE) |
| static int | [**CV\_DIST\_C**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_DIST_C) |
| static int | [**CV\_DIST\_FAIR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_DIST_FAIR) |
| static int | [**CV\_DIST\_HUBER**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_DIST_HUBER) |
| static int | [**CV\_DIST\_L1**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_DIST_L1) |
| static int | [**CV\_DIST\_L12**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_DIST_L12) |
| static int | [**CV\_DIST\_L2**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_DIST_L2) |
| static int | [**CV\_DIST\_LABEL\_CCOMP**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_DIST_LABEL_CCOMP) |
| static int | [**CV\_DIST\_LABEL\_PIXEL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_DIST_LABEL_PIXEL) |
| static int | [**CV\_DIST\_MASK\_3**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_DIST_MASK_3) |
| static int | [**CV\_DIST\_MASK\_5**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_DIST_MASK_5) |
| static int | [**CV\_DIST\_MASK\_PRECISE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_DIST_MASK_PRECISE) |
| static int | [**CV\_DIST\_USER**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_DIST_USER) |
| static int | [**CV\_DIST\_WELSCH**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_DIST_WELSCH) |
| static int | [**CV\_GAUSSIAN**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_GAUSSIAN) |
| static int | [**CV\_GAUSSIAN\_5x5**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_GAUSSIAN_5x5) |
| static int | [**CV\_HOUGH\_GRADIENT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_HOUGH_GRADIENT) |
| static int | [**CV\_HOUGH\_MULTI\_SCALE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_HOUGH_MULTI_SCALE) |
| static int | [**CV\_HOUGH\_PROBABILISTIC**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_HOUGH_PROBABILISTIC) |
| static int | [**CV\_HOUGH\_STANDARD**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_HOUGH_STANDARD) |
| static int | [**CV\_LINK\_RUNS**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_LINK_RUNS) |
| static int | [**CV\_MAX\_SOBEL\_KSIZE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_MAX_SOBEL_KSIZE) |
| static int | [**CV\_MEDIAN**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_MEDIAN) |
| static int | [**CV\_mRGBA2RGBA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_mRGBA2RGBA) |
| static int | [**CV\_POLY\_APPROX\_DP**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_POLY_APPROX_DP) |
| static int | [**CV\_RGBA2mRGBA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_RGBA2mRGBA) |
| static int | [**CV\_SCHARR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_SCHARR) |
| static int | [**CV\_SHAPE\_CROSS**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_SHAPE_CROSS) |
| static int | [**CV\_SHAPE\_CUSTOM**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_SHAPE_CUSTOM) |
| static int | [**CV\_SHAPE\_ELLIPSE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_SHAPE_ELLIPSE) |
| static int | [**CV\_SHAPE\_RECT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_SHAPE_RECT) |
| static int | [**CV\_WARP\_FILL\_OUTLIERS**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_WARP_FILL_OUTLIERS) |
| static int | [**CV\_WARP\_INVERSE\_MAP**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#CV_WARP_INVERSE_MAP) |
| static int | [**DIST\_C**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#DIST_C) |
| static int | [**DIST\_FAIR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#DIST_FAIR) |
| static int | [**DIST\_HUBER**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#DIST_HUBER) |
| static int | [**DIST\_L1**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#DIST_L1) |
| static int | [**DIST\_L12**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#DIST_L12) |
| static int | [**DIST\_L2**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#DIST_L2) |
| static int | [**DIST\_LABEL\_CCOMP**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#DIST_LABEL_CCOMP) |
| static int | [**DIST\_LABEL\_PIXEL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#DIST_LABEL_PIXEL) |
| static int | [**DIST\_MASK\_3**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#DIST_MASK_3) |
| static int | [**DIST\_MASK\_5**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#DIST_MASK_5) |
| static int | [**DIST\_MASK\_PRECISE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#DIST_MASK_PRECISE) |
| static int | [**DIST\_USER**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#DIST_USER) |
| static int | [**DIST\_WELSCH**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#DIST_WELSCH) |
| static int | [**FLOODFILL\_FIXED\_RANGE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#FLOODFILL_FIXED_RANGE) |
| static int | [**FLOODFILL\_MASK\_ONLY**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#FLOODFILL_MASK_ONLY) |
| static int | [**GC\_BGD**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#GC_BGD) |
| static int | [**GC\_EVAL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#GC_EVAL) |
| static int | [**GC\_EVAL\_FREEZE\_MODEL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#GC_EVAL_FREEZE_MODEL) |
| static int | [**GC\_FGD**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#GC_FGD) |
| static int | [**GC\_INIT\_WITH\_MASK**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#GC_INIT_WITH_MASK) |
| static int | [**GC\_INIT\_WITH\_RECT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#GC_INIT_WITH_RECT) |
| static int | [**GC\_PR\_BGD**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#GC_PR_BGD) |
| static int | [**GC\_PR\_FGD**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#GC_PR_FGD) |
| static int | [**HISTCMP\_BHATTACHARYYA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HISTCMP_BHATTACHARYYA) |
| static int | [**HISTCMP\_CHISQR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HISTCMP_CHISQR) |
| static int | [**HISTCMP\_CHISQR\_ALT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HISTCMP_CHISQR_ALT) |
| static int | [**HISTCMP\_CORREL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HISTCMP_CORREL) |
| static int | [**HISTCMP\_HELLINGER**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HISTCMP_HELLINGER) |
| static int | [**HISTCMP\_INTERSECT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HISTCMP_INTERSECT) |
| static int | [**HISTCMP\_KL\_DIV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HISTCMP_KL_DIV) |
| static int | [**HOUGH\_GRADIENT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HOUGH_GRADIENT) |
| static int | [**HOUGH\_MULTI\_SCALE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HOUGH_MULTI_SCALE) |
| static int | [**HOUGH\_PROBABILISTIC**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HOUGH_PROBABILISTIC) |
| static int | [**HOUGH\_STANDARD**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HOUGH_STANDARD) |
| static int | [**INTER\_AREA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#INTER_AREA) |
| static int | [**INTER\_BITS**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#INTER_BITS) |
| static int | [**INTER\_BITS2**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#INTER_BITS2) |
| static int | [**INTER\_CUBIC**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#INTER_CUBIC) |
| static int | [**INTER\_LANCZOS4**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#INTER_LANCZOS4) |
| static int | [**INTER\_LINEAR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#INTER_LINEAR) |
| static int | [**INTER\_LINEAR\_EXACT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#INTER_LINEAR_EXACT) |
| static int | [**INTER\_MAX**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#INTER_MAX) |
| static int | [**INTER\_NEAREST**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#INTER_NEAREST) |
| static int | [**INTER\_NEAREST\_EXACT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#INTER_NEAREST_EXACT) |
| static int | [**INTER\_TAB\_SIZE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#INTER_TAB_SIZE) |
| static int | [**INTER\_TAB\_SIZE2**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#INTER_TAB_SIZE2) |
| static int | [**INTERSECT\_FULL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#INTERSECT_FULL) |
| static int | [**INTERSECT\_NONE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#INTERSECT_NONE) |
| static int | [**INTERSECT\_PARTIAL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#INTERSECT_PARTIAL) |
| static int | [**LINE\_4**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#LINE_4) |
| static int | [**LINE\_8**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#LINE_8) |
| static int | [**LINE\_AA**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#LINE_AA) |
| static int | [**LSD\_REFINE\_ADV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#LSD_REFINE_ADV) |
| static int | [**LSD\_REFINE\_NONE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#LSD_REFINE_NONE) |
| static int | [**LSD\_REFINE\_STD**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#LSD_REFINE_STD) |
| static int | [**MARKER\_CROSS**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#MARKER_CROSS) |
| static int | [**MARKER\_DIAMOND**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#MARKER_DIAMOND) |
| static int | [**MARKER\_SQUARE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#MARKER_SQUARE) |
| static int | [**MARKER\_STAR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#MARKER_STAR) |
| static int | [**MARKER\_TILTED\_CROSS**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#MARKER_TILTED_CROSS) |
| static int | [**MARKER\_TRIANGLE\_DOWN**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#MARKER_TRIANGLE_DOWN) |
| static int | [**MARKER\_TRIANGLE\_UP**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#MARKER_TRIANGLE_UP) |
| static int | [**MORPH\_BLACKHAT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#MORPH_BLACKHAT) |
| static int | [**MORPH\_CLOSE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#MORPH_CLOSE) |
| static int | [**MORPH\_CROSS**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#MORPH_CROSS) |
| static int | [**MORPH\_DILATE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#MORPH_DILATE) |
| static int | [**MORPH\_ELLIPSE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#MORPH_ELLIPSE) |
| static int | [**MORPH\_ERODE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#MORPH_ERODE) |
| static int | [**MORPH\_GRADIENT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#MORPH_GRADIENT) |
| static int | [**MORPH\_HITMISS**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#MORPH_HITMISS) |
| static int | [**MORPH\_OPEN**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#MORPH_OPEN) |
| static int | [**MORPH\_RECT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#MORPH_RECT) |
| static int | [**MORPH\_TOPHAT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#MORPH_TOPHAT) |
| static int | [**PROJ\_SPHERICAL\_EQRECT**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#PROJ_SPHERICAL_EQRECT) |
| static int | [**PROJ\_SPHERICAL\_ORTHO**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#PROJ_SPHERICAL_ORTHO) |
| static int | [**RETR\_CCOMP**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#RETR_CCOMP) |
| static int | [**RETR\_EXTERNAL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#RETR_EXTERNAL) |
| static int | [**RETR\_FLOODFILL**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#RETR_FLOODFILL) |
| static int | [**RETR\_LIST**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#RETR_LIST) |
| static int | [**RETR\_TREE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#RETR_TREE) |
| static int | [**THRESH\_BINARY**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#THRESH_BINARY) |
| static int | [**THRESH\_BINARY\_INV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#THRESH_BINARY_INV) |
| static int | [**THRESH\_MASK**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#THRESH_MASK) |
| static int | [**THRESH\_OTSU**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#THRESH_OTSU) |
| static int | [**THRESH\_TOZERO**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#THRESH_TOZERO) |
| static int | [**THRESH\_TOZERO\_INV**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#THRESH_TOZERO_INV) |
| static int | [**THRESH\_TRIANGLE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#THRESH_TRIANGLE) |
| static int | [**THRESH\_TRUNC**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#THRESH_TRUNC) |
| static int | [**TM\_CCOEFF**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#TM_CCOEFF) |
| static int | [**TM\_CCOEFF\_NORMED**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#TM_CCOEFF_NORMED) |
| static int | [**TM\_CCORR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#TM_CCORR) |
| static int | [**TM\_CCORR\_NORMED**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#TM_CCORR_NORMED) |
| static int | [**TM\_SQDIFF**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#TM_SQDIFF) |
| static int | [**TM\_SQDIFF\_NORMED**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#TM_SQDIFF_NORMED) |
| static int | [**WARP\_FILL\_OUTLIERS**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#WARP_FILL_OUTLIERS) |
| static int | [**WARP\_INVERSE\_MAP**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#WARP_INVERSE_MAP) |
| static int | [**WARP\_POLAR\_LINEAR**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#WARP_POLAR_LINEAR) |
| static int | [**WARP\_POLAR\_LOG**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#WARP_POLAR_LOG) |

### Constructor SummaryConstructors

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

### Method SummaryMethods

| Modifier and Type | Method and Description |
| --- | --- |
| static void | [**accumulate**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#accumulate(org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Adds an image to the accumulator image. |
| static void | [**accumulate**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#accumulate(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask) Adds an image to the accumulator image. |
| static void | [**accumulateProduct**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#accumulateProduct(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) src2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Adds the per-element product of two input images to the accumulator image. |
| static void | [**accumulateProduct**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#accumulateProduct(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) src2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask) Adds the per-element product of two input images to the accumulator image. |
| static void | [**accumulateSquare**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#accumulateSquare(org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Adds the square of a source image to the accumulator image. |
| static void | [**accumulateSquare**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#accumulateSquare(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask) Adds the square of a source image to the accumulator image. |
| static void | [**accumulateWeighted**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#accumulateWeighted(org.opencv.core.Mat,%20org.opencv.core.Mat,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, double alpha) Updates a running average. |
| static void | [**accumulateWeighted**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#accumulateWeighted(org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, double alpha, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask) Updates a running average. |
| static void | [**adaptiveThreshold**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#adaptiveThreshold(org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20int,%20int,%20int,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, double maxValue, int adaptiveMethod, int thresholdType, int blockSize, double C) Applies an adaptive threshold to an array. |
| static void | [**applyColorMap**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#applyColorMap(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int colormap) Applies a GNU Octave/MATLAB equivalent colormap on a given image. |
| static void | [**applyColorMap**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#applyColorMap(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) userColor) Applies a user colormap on a given image. |
| static void | [**approxPolyDP**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#approxPolyDP(org.opencv.core.MatOfPoint2f,%20org.opencv.core.MatOfPoint2f,%20double,%20boolean))([MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) curve, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) approxCurve, double epsilon, boolean closed) Approximates a polygonal curve(s) with the specified precision. |
| static double | [**arcLength**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#arcLength(org.opencv.core.MatOfPoint2f,%20boolean))([MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) curve, boolean closed) Calculates a contour perimeter or a curve length. |
| static void | [**arrowedLine**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#arrowedLine(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Point,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Draws a arrow segment pointing from the first point to the second one. |
| static void | [**arrowedLine**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#arrowedLine(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Point,%20org.opencv.core.Scalar,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness) Draws a arrow segment pointing from the first point to the second one. |
| static void | [**arrowedLine**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#arrowedLine(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Point,%20org.opencv.core.Scalar,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int line\_type) Draws a arrow segment pointing from the first point to the second one. |
| static void | [**arrowedLine**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#arrowedLine(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Point,%20org.opencv.core.Scalar,%20int,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int line\_type, int shift) Draws a arrow segment pointing from the first point to the second one. |
| static void | [**arrowedLine**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#arrowedLine(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Point,%20org.opencv.core.Scalar,%20int,%20int,%20int,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int line\_type, int shift, double tipLength) Draws a arrow segment pointing from the first point to the second one. |
| static void | [**bilateralFilter**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#bilateralFilter(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20double,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int d, double sigmaColor, double sigmaSpace) Applies the bilateral filter to an image. |
| static void | [**bilateralFilter**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#bilateralFilter(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20double,%20double,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int d, double sigmaColor, double sigmaSpace, int borderType) Applies the bilateral filter to an image. |
| static void | [**blendLinear**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#blendLinear(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) src2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) weights1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) weights2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) |
| static void | [**blur**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#blur(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize) Blurs an image using the normalized box filter. |
| static void | [**blur**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#blur(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20org.opencv.core.Point))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor) Blurs an image using the normalized box filter. |
| static void | [**blur**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#blur(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20org.opencv.core.Point,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int borderType) Blurs an image using the normalized box filter. |
| static [Rect](http://docs.google.com/org/opencv/core/Rect.html) | [**boundingRect**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#boundingRect(org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) array) Calculates the up-right bounding rectangle of a point set or non-zero pixels of gray-scale image. |
| static void | [**boxFilter**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#boxFilter(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Size))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize) Blurs an image using the box filter. |
| static void | [**boxFilter**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#boxFilter(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Size,%20org.opencv.core.Point))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor) Blurs an image using the box filter. |
| static void | [**boxFilter**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#boxFilter(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Size,%20org.opencv.core.Point,%20boolean))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, boolean normalize) Blurs an image using the box filter. |
| static void | [**boxFilter**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#boxFilter(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Size,%20org.opencv.core.Point,%20boolean,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, boolean normalize, int borderType) Blurs an image using the box filter. |
| static void | [**boxPoints**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#boxPoints(org.opencv.core.RotatedRect,%20org.opencv.core.Mat))([RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) box, [Mat](http://docs.google.com/org/opencv/core/Mat.html) points) Finds the four vertices of a rotated rect. |
| static void | [**calcBackProject**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#calcBackProject(java.util.List,%20org.opencv.core.MatOfInt,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.MatOfFloat,%20double))(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> images, [MatOfInt](http://docs.google.com/org/opencv/core/MatOfInt.html) channels, [Mat](http://docs.google.com/org/opencv/core/Mat.html) hist, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) ranges, double scale) |
| static void | [**calcHist**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#calcHist(java.util.List,%20org.opencv.core.MatOfInt,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.MatOfInt,%20org.opencv.core.MatOfFloat))(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> images, [MatOfInt](http://docs.google.com/org/opencv/core/MatOfInt.html) channels, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) hist, [MatOfInt](http://docs.google.com/org/opencv/core/MatOfInt.html) histSize, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) ranges) |
| static void | [**calcHist**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#calcHist(java.util.List,%20org.opencv.core.MatOfInt,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.MatOfInt,%20org.opencv.core.MatOfFloat,%20boolean))(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> images, [MatOfInt](http://docs.google.com/org/opencv/core/MatOfInt.html) channels, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) hist, [MatOfInt](http://docs.google.com/org/opencv/core/MatOfInt.html) histSize, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) ranges, boolean accumulate) |
| static void | [**Canny**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#Canny(org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) edges, double threshold1, double threshold2) Finds edges in an image using the Canny algorithm CITE: Canny86 . |
| static void | [**Canny**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#Canny(org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20double,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) edges, double threshold1, double threshold2, int apertureSize) Finds edges in an image using the Canny algorithm CITE: Canny86 . |
| static void | [**Canny**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#Canny(org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20double,%20int,%20boolean))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) edges, double threshold1, double threshold2, int apertureSize, boolean L2gradient) Finds edges in an image using the Canny algorithm CITE: Canny86 . |
| static void | [**Canny**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#Canny(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) dx, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dy, [Mat](http://docs.google.com/org/opencv/core/Mat.html) edges, double threshold1, double threshold2) \overload Finds edges in an image using the Canny algorithm with custom image gradient. |
| static void | [**Canny**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#Canny(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20double,%20boolean))([Mat](http://docs.google.com/org/opencv/core/Mat.html) dx, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dy, [Mat](http://docs.google.com/org/opencv/core/Mat.html) edges, double threshold1, double threshold2, boolean L2gradient) \overload Finds edges in an image using the Canny algorithm with custom image gradient. |
| static void | [**circle**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#circle(org.opencv.core.Mat,%20org.opencv.core.Point,%20int,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) center, int radius, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Draws a circle. |
| static void | [**circle**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#circle(org.opencv.core.Mat,%20org.opencv.core.Point,%20int,%20org.opencv.core.Scalar,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) center, int radius, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness) Draws a circle. |
| static void | [**circle**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#circle(org.opencv.core.Mat,%20org.opencv.core.Point,%20int,%20org.opencv.core.Scalar,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) center, int radius, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType) Draws a circle. |
| static void | [**circle**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#circle(org.opencv.core.Mat,%20org.opencv.core.Point,%20int,%20org.opencv.core.Scalar,%20int,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) center, int radius, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType, int shift) Draws a circle. |
| static boolean | [**clipLine**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#clipLine(org.opencv.core.Rect,%20org.opencv.core.Point,%20org.opencv.core.Point))([Rect](http://docs.google.com/org/opencv/core/Rect.html) imgRect, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2) |
| static double | [**compareHist**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#compareHist(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) H1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) H2, int method) Compares two histograms. |
| static int | [**connectedComponents**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#connectedComponents(org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels) |
| static int | [**connectedComponents**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#connectedComponents(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels, int connectivity) |
| static int | [**connectedComponents**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#connectedComponents(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels, int connectivity, int ltype) |
| static int | [**connectedComponentsWithAlgorithm**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#connectedComponentsWithAlgorithm(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels, int connectivity, int ltype, int ccltype) computes the connected components labeled image of boolean image image with 4 or 8 way connectivity - returns N, the total number of labels [0, N-1] where 0 represents the background label. |
| static int | [**connectedComponentsWithStats**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#connectedComponentsWithStats(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels, [Mat](http://docs.google.com/org/opencv/core/Mat.html) stats, [Mat](http://docs.google.com/org/opencv/core/Mat.html) centroids) |
| static int | [**connectedComponentsWithStats**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#connectedComponentsWithStats(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels, [Mat](http://docs.google.com/org/opencv/core/Mat.html) stats, [Mat](http://docs.google.com/org/opencv/core/Mat.html) centroids, int connectivity) |
| static int | [**connectedComponentsWithStats**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#connectedComponentsWithStats(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels, [Mat](http://docs.google.com/org/opencv/core/Mat.html) stats, [Mat](http://docs.google.com/org/opencv/core/Mat.html) centroids, int connectivity, int ltype) |
| static int | [**connectedComponentsWithStatsWithAlgorithm**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#connectedComponentsWithStatsWithAlgorithm(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels, [Mat](http://docs.google.com/org/opencv/core/Mat.html) stats, [Mat](http://docs.google.com/org/opencv/core/Mat.html) centroids, int connectivity, int ltype, int ccltype) computes the connected components labeled image of boolean image and also produces a statistics output for each label image with 4 or 8 way connectivity - returns N, the total number of labels [0, N-1] where 0 represents the background label. |
| static double | [**contourArea**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#contourArea(org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) contour) Calculates a contour area. |
| static double | [**contourArea**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#contourArea(org.opencv.core.Mat,%20boolean))([Mat](http://docs.google.com/org/opencv/core/Mat.html) contour, boolean oriented) Calculates a contour area. |
| static void | [**convertMaps**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#convertMaps(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) map1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dstmap1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dstmap2, int dstmap1type) Converts image transformation maps from one representation to another. |
| static void | [**convertMaps**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#convertMaps(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20boolean))([Mat](http://docs.google.com/org/opencv/core/Mat.html) map1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dstmap1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dstmap2, int dstmap1type, boolean nninterpolation) Converts image transformation maps from one representation to another. |
| static void | [**convexHull**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#convexHull(org.opencv.core.MatOfPoint,%20org.opencv.core.MatOfInt))([MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) points, [MatOfInt](http://docs.google.com/org/opencv/core/MatOfInt.html) hull) Finds the convex hull of a point set. |
| static void | [**convexHull**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#convexHull(org.opencv.core.MatOfPoint,%20org.opencv.core.MatOfInt,%20boolean))([MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) points, [MatOfInt](http://docs.google.com/org/opencv/core/MatOfInt.html) hull, boolean clockwise) Finds the convex hull of a point set. |
| static void | [**convexityDefects**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#convexityDefects(org.opencv.core.MatOfPoint,%20org.opencv.core.MatOfInt,%20org.opencv.core.MatOfInt4))([MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) contour, [MatOfInt](http://docs.google.com/org/opencv/core/MatOfInt.html) convexhull, [MatOfInt4](http://docs.google.com/org/opencv/core/MatOfInt4.html) convexityDefects) Finds the convexity defects of a contour. |
| static void | [**cornerEigenValsAndVecs**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#cornerEigenValsAndVecs(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int blockSize, int ksize) Calculates eigenvalues and eigenvectors of image blocks for corner detection. |
| static void | [**cornerEigenValsAndVecs**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#cornerEigenValsAndVecs(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int blockSize, int ksize, int borderType) Calculates eigenvalues and eigenvectors of image blocks for corner detection. |
| static void | [**cornerHarris**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#cornerHarris(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int blockSize, int ksize, double k) Harris corner detector. |
| static void | [**cornerHarris**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#cornerHarris(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%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, int blockSize, int ksize, double k, int borderType) Harris corner detector. |
| static void | [**cornerMinEigenVal**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#cornerMinEigenVal(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int blockSize) Calculates the minimal eigenvalue of gradient matrices for corner detection. |
| static void | [**cornerMinEigenVal**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#cornerMinEigenVal(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int blockSize, int ksize) Calculates the minimal eigenvalue of gradient matrices for corner detection. |
| static void | [**cornerMinEigenVal**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#cornerMinEigenVal(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int blockSize, int ksize, int borderType) Calculates the minimal eigenvalue of gradient matrices for corner detection. |
| static void | [**cornerSubPix**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#cornerSubPix(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20org.opencv.core.Size,%20org.opencv.core.TermCriteria))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) corners, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, [Size](http://docs.google.com/org/opencv/core/Size.html) zeroZone, [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) criteria) Refines the corner locations. |
| static [CLAHE](http://docs.google.com/org/opencv/imgproc/CLAHE.html) | [**createCLAHE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#createCLAHE())() Creates a smart pointer to a cv::CLAHE class and initializes it. |
| static [CLAHE](http://docs.google.com/org/opencv/imgproc/CLAHE.html) | [**createCLAHE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#createCLAHE(double))(double clipLimit) Creates a smart pointer to a cv::CLAHE class and initializes it. |
| static [CLAHE](http://docs.google.com/org/opencv/imgproc/CLAHE.html) | [**createCLAHE**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#createCLAHE(double,%20org.opencv.core.Size))(double clipLimit, [Size](http://docs.google.com/org/opencv/core/Size.html) tileGridSize) Creates a smart pointer to a cv::CLAHE class and initializes it. |
| static [GeneralizedHoughBallard](http://docs.google.com/org/opencv/imgproc/GeneralizedHoughBallard.html) | [**createGeneralizedHoughBallard**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#createGeneralizedHoughBallard())() Creates a smart pointer to a cv::GeneralizedHoughBallard class and initializes it. |
| static [GeneralizedHoughGuil](http://docs.google.com/org/opencv/imgproc/GeneralizedHoughGuil.html) | [**createGeneralizedHoughGuil**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#createGeneralizedHoughGuil())() Creates a smart pointer to a cv::GeneralizedHoughGuil class and initializes it. |
| static void | [**createHanningWindow**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#createHanningWindow(org.opencv.core.Mat,%20org.opencv.core.Size,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int type) This function computes a Hanning window coefficients in two dimensions. |
| static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) | [**createLineSegmentDetector**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#createLineSegmentDetector())() Creates a smart pointer to a LineSegmentDetector object and initializes it. |
| static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) | [**createLineSegmentDetector**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#createLineSegmentDetector(int))(int \_refine) Creates a smart pointer to a LineSegmentDetector object and initializes it. |
| static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) | [**createLineSegmentDetector**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#createLineSegmentDetector(int,%20double))(int \_refine, double \_scale) Creates a smart pointer to a LineSegmentDetector object and initializes it. |
| static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) | [**createLineSegmentDetector**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#createLineSegmentDetector(int,%20double,%20double))(int \_refine, double \_scale, double \_sigma\_scale) Creates a smart pointer to a LineSegmentDetector object and initializes it. |
| static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) | [**createLineSegmentDetector**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#createLineSegmentDetector(int,%20double,%20double,%20double))(int \_refine, double \_scale, double \_sigma\_scale, double \_quant) Creates a smart pointer to a LineSegmentDetector object and initializes it. |
| static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) | [**createLineSegmentDetector**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#createLineSegmentDetector(int,%20double,%20double,%20double,%20double))(int \_refine, double \_scale, double \_sigma\_scale, double \_quant, double \_ang\_th) Creates a smart pointer to a LineSegmentDetector object and initializes it. |
| static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) | [**createLineSegmentDetector**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#createLineSegmentDetector(int,%20double,%20double,%20double,%20double,%20double))(int \_refine, double \_scale, double \_sigma\_scale, double \_quant, double \_ang\_th, double \_log\_eps) Creates a smart pointer to a LineSegmentDetector object and initializes it. |
| static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) | [**createLineSegmentDetector**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#createLineSegmentDetector(int,%20double,%20double,%20double,%20double,%20double,%20double))(int \_refine, double \_scale, double \_sigma\_scale, double \_quant, double \_ang\_th, double \_log\_eps, double \_density\_th) Creates a smart pointer to a LineSegmentDetector object and initializes it. |
| static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) | [**createLineSegmentDetector**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#createLineSegmentDetector(int,%20double,%20double,%20double,%20double,%20double,%20double,%20int))(int \_refine, double \_scale, double \_sigma\_scale, double \_quant, double \_ang\_th, double \_log\_eps, double \_density\_th, int \_n\_bins) Creates a smart pointer to a LineSegmentDetector object and initializes it. |
| static void | [**cvtColor**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#cvtColor(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int code) Converts an image from one color space to another. |
| static void | [**cvtColor**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#cvtColor(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int code, int dstCn) Converts an image from one color space to another. |
| static void | [**cvtColorTwoPlane**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#cvtColorTwoPlane(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) src2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int code) Converts an image from one color space to another where the source image is stored in two planes. |
| static void | [**demosaicing**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#demosaicing(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int code) main function for all demosaicing processes |
| static void | [**demosaicing**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#demosaicing(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int code, int dstCn) main function for all demosaicing processes |
| static void | [**dilate**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#dilate(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel) Dilates an image by using a specific structuring element. |
| static void | [**dilate**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#dilate(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Point))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor) Dilates an image by using a specific structuring element. |
| static void | [**dilate**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#dilate(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Point,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int iterations) Dilates an image by using a specific structuring element. |
| static void | [**dilate**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#dilate(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Point,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int iterations, int borderType) Dilates an image by using a specific structuring element. |
| static void | [**dilate**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#dilate(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Point,%20int,%20int,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int iterations, int borderType, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) borderValue) Dilates an image by using a specific structuring element. |
| static void | [**distanceTransform**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#distanceTransform(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int distanceType, int maskSize) |
| static void | [**distanceTransform**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#distanceTransform(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int distanceType, int maskSize, int dstType) |
| static void | [**distanceTransformWithLabels**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#distanceTransformWithLabels(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels, int distanceType, int maskSize) Calculates the distance to the closest zero pixel for each pixel of the source image. |
| static void | [**distanceTransformWithLabels**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#distanceTransformWithLabels(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels, int distanceType, int maskSize, int labelType) Calculates the distance to the closest zero pixel for each pixel of the source image. |
| static void | [**drawContours**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#drawContours(org.opencv.core.Mat,%20java.util.List,%20int,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> contours, int contourIdx, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Draws contours outlines or filled contours. |
| static void | [**drawContours**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#drawContours(org.opencv.core.Mat,%20java.util.List,%20int,%20org.opencv.core.Scalar,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> contours, int contourIdx, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness) Draws contours outlines or filled contours. |
| static void | [**drawContours**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#drawContours(org.opencv.core.Mat,%20java.util.List,%20int,%20org.opencv.core.Scalar,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> contours, int contourIdx, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType) Draws contours outlines or filled contours. |
| static void | [**drawContours**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#drawContours(org.opencv.core.Mat,%20java.util.List,%20int,%20org.opencv.core.Scalar,%20int,%20int,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> contours, int contourIdx, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType, [Mat](http://docs.google.com/org/opencv/core/Mat.html) hierarchy) Draws contours outlines or filled contours. |
| static void | [**drawContours**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#drawContours(org.opencv.core.Mat,%20java.util.List,%20int,%20org.opencv.core.Scalar,%20int,%20int,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> contours, int contourIdx, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType, [Mat](http://docs.google.com/org/opencv/core/Mat.html) hierarchy, int maxLevel) Draws contours outlines or filled contours. |
| static void | [**drawContours**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#drawContours(org.opencv.core.Mat,%20java.util.List,%20int,%20org.opencv.core.Scalar,%20int,%20int,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Point))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> contours, int contourIdx, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType, [Mat](http://docs.google.com/org/opencv/core/Mat.html) hierarchy, int maxLevel, [Point](http://docs.google.com/org/opencv/core/Point.html) offset) Draws contours outlines or filled contours. |
| static void | [**drawMarker**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#drawMarker(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) position, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Draws a marker on a predefined position in an image. |
| static void | [**drawMarker**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#drawMarker(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Scalar,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) position, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int markerType) Draws a marker on a predefined position in an image. |
| static void | [**drawMarker**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#drawMarker(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Scalar,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) position, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int markerType, int markerSize) Draws a marker on a predefined position in an image. |
| static void | [**drawMarker**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#drawMarker(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Scalar,%20int,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) position, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int markerType, int markerSize, int thickness) Draws a marker on a predefined position in an image. |
| static void | [**drawMarker**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#drawMarker(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Scalar,%20int,%20int,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) position, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int markerType, int markerSize, int thickness, int line\_type) Draws a marker on a predefined position in an image. |
| static void | [**ellipse**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#ellipse(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Size,%20double,%20double,%20double,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) center, [Size](http://docs.google.com/org/opencv/core/Size.html) axes, double angle, double startAngle, double endAngle, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Draws a simple or thick elliptic arc or fills an ellipse sector. |
| static void | [**ellipse**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#ellipse(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Size,%20double,%20double,%20double,%20org.opencv.core.Scalar,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) center, [Size](http://docs.google.com/org/opencv/core/Size.html) axes, double angle, double startAngle, double endAngle, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness) Draws a simple or thick elliptic arc or fills an ellipse sector. |
| static void | [**ellipse**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#ellipse(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Size,%20double,%20double,%20double,%20org.opencv.core.Scalar,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) center, [Size](http://docs.google.com/org/opencv/core/Size.html) axes, double angle, double startAngle, double endAngle, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType) Draws a simple or thick elliptic arc or fills an ellipse sector. |
| static void | [**ellipse**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#ellipse(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Size,%20double,%20double,%20double,%20org.opencv.core.Scalar,%20int,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) center, [Size](http://docs.google.com/org/opencv/core/Size.html) axes, double angle, double startAngle, double endAngle, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType, int shift) Draws a simple or thick elliptic arc or fills an ellipse sector. |
| static void | [**ellipse**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#ellipse(org.opencv.core.Mat,%20org.opencv.core.RotatedRect,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) box, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) |
| static void | [**ellipse**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#ellipse(org.opencv.core.Mat,%20org.opencv.core.RotatedRect,%20org.opencv.core.Scalar,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) box, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness) |
| static void | [**ellipse**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#ellipse(org.opencv.core.Mat,%20org.opencv.core.RotatedRect,%20org.opencv.core.Scalar,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) box, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType) |
| static void | [**ellipse2Poly**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#ellipse2Poly(org.opencv.core.Point,%20org.opencv.core.Size,%20int,%20int,%20int,%20int,%20org.opencv.core.MatOfPoint))([Point](http://docs.google.com/org/opencv/core/Point.html) center, [Size](http://docs.google.com/org/opencv/core/Size.html) axes, int angle, int arcStart, int arcEnd, int delta, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) pts) Approximates an elliptic arc with a polyline. |
| static float | [**EMD**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#EMD(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) signature1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) signature2, int distType) Computes the "minimal work" distance between two weighted point configurations. |
| static float | [**EMD**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#EMD(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) signature1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) signature2, int distType, [Mat](http://docs.google.com/org/opencv/core/Mat.html) cost) Computes the "minimal work" distance between two weighted point configurations. |
| static float | [**EMD**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#EMD(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) signature1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) signature2, int distType, [Mat](http://docs.google.com/org/opencv/core/Mat.html) cost, [Mat](http://docs.google.com/org/opencv/core/Mat.html) flow) Computes the "minimal work" distance between two weighted point configurations. |
| static void | [**equalizeHist**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#equalizeHist(org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Equalizes the histogram of a grayscale image. |
| static void | [**erode**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#erode(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel) Erodes an image by using a specific structuring element. |
| static void | [**erode**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#erode(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Point))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor) Erodes an image by using a specific structuring element. |
| static void | [**erode**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#erode(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Point,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int iterations) Erodes an image by using a specific structuring element. |
| static void | [**erode**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#erode(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Point,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int iterations, int borderType) Erodes an image by using a specific structuring element. |
| static void | [**erode**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#erode(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Point,%20int,%20int,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int iterations, int borderType, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) borderValue) Erodes an image by using a specific structuring element. |
| static void | [**fillConvexPoly**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#fillConvexPoly(org.opencv.core.Mat,%20org.opencv.core.MatOfPoint,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) points, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Fills a convex polygon. |
| static void | [**fillConvexPoly**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#fillConvexPoly(org.opencv.core.Mat,%20org.opencv.core.MatOfPoint,%20org.opencv.core.Scalar,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) points, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int lineType) Fills a convex polygon. |
| static void | [**fillConvexPoly**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#fillConvexPoly(org.opencv.core.Mat,%20org.opencv.core.MatOfPoint,%20org.opencv.core.Scalar,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) points, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int lineType, int shift) Fills a convex polygon. |
| static void | [**fillPoly**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#fillPoly(org.opencv.core.Mat,%20java.util.List,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> pts, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Fills the area bounded by one or more polygons. |
| static void | [**fillPoly**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#fillPoly(org.opencv.core.Mat,%20java.util.List,%20org.opencv.core.Scalar,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> pts, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int lineType) Fills the area bounded by one or more polygons. |
| static void | [**fillPoly**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#fillPoly(org.opencv.core.Mat,%20java.util.List,%20org.opencv.core.Scalar,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> pts, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int lineType, int shift) Fills the area bounded by one or more polygons. |
| static void | [**fillPoly**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#fillPoly(org.opencv.core.Mat,%20java.util.List,%20org.opencv.core.Scalar,%20int,%20int,%20org.opencv.core.Point))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> pts, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int lineType, int shift, [Point](http://docs.google.com/org/opencv/core/Point.html) offset) Fills the area bounded by one or more polygons. |
| static void | [**filter2D**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#filter2D(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel) Convolves an image with the kernel. |
| static void | [**filter2D**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#filter2D(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20org.opencv.core.Point))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor) Convolves an image with the kernel. |
| static void | [**filter2D**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#filter2D(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20org.opencv.core.Point,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, double delta) Convolves an image with the kernel. |
| static void | [**filter2D**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#filter2D(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20org.opencv.core.Point,%20double,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, double delta, int borderType) Convolves an image with the kernel. |
| static void | [**findContours**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#findContours(org.opencv.core.Mat,%20java.util.List,%20org.opencv.core.Mat,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> contours, [Mat](http://docs.google.com/org/opencv/core/Mat.html) hierarchy, int mode, int method) Finds contours in a binary image. |
| static void | [**findContours**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#findContours(org.opencv.core.Mat,%20java.util.List,%20org.opencv.core.Mat,%20int,%20int,%20org.opencv.core.Point))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> contours, [Mat](http://docs.google.com/org/opencv/core/Mat.html) hierarchy, int mode, int method, [Point](http://docs.google.com/org/opencv/core/Point.html) offset) Finds contours in a binary image. |
| static [RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) | [**fitEllipse**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#fitEllipse(org.opencv.core.MatOfPoint2f))([MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) points) Fits an ellipse around a set of 2D points. |
| static [RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) | [**fitEllipseAMS**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#fitEllipseAMS(org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) points) Fits an ellipse around a set of 2D points. |
| static [RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) | [**fitEllipseDirect**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#fitEllipseDirect(org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) points) Fits an ellipse around a set of 2D points. |
| static void | [**fitLine**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#fitLine(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20double,%20double,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) points, [Mat](http://docs.google.com/org/opencv/core/Mat.html) line, int distType, double param, double reps, double aeps) Fits a line to a 2D or 3D point set. |
| static int | [**floodFill**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#floodFill(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Point](http://docs.google.com/org/opencv/core/Point.html) seedPoint, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) newVal) Fills a connected component with the given color. |
| static int | [**floodFill**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#floodFill(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Scalar,%20org.opencv.core.Rect))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Point](http://docs.google.com/org/opencv/core/Point.html) seedPoint, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) newVal, [Rect](http://docs.google.com/org/opencv/core/Rect.html) rect) Fills a connected component with the given color. |
| static int | [**floodFill**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#floodFill(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Scalar,%20org.opencv.core.Rect,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Point](http://docs.google.com/org/opencv/core/Point.html) seedPoint, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) newVal, [Rect](http://docs.google.com/org/opencv/core/Rect.html) rect, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) loDiff) Fills a connected component with the given color. |
| static int | [**floodFill**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#floodFill(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Scalar,%20org.opencv.core.Rect,%20org.opencv.core.Scalar,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Point](http://docs.google.com/org/opencv/core/Point.html) seedPoint, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) newVal, [Rect](http://docs.google.com/org/opencv/core/Rect.html) rect, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) loDiff, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) upDiff) Fills a connected component with the given color. |
| static int | [**floodFill**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#floodFill(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Scalar,%20org.opencv.core.Rect,%20org.opencv.core.Scalar,%20org.opencv.core.Scalar,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Point](http://docs.google.com/org/opencv/core/Point.html) seedPoint, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) newVal, [Rect](http://docs.google.com/org/opencv/core/Rect.html) rect, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) loDiff, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) upDiff, int flags) Fills a connected component with the given color. |
| static void | [**GaussianBlur**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#GaussianBlur(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, double sigmaX) Blurs an image using a Gaussian filter. |
| static void | [**GaussianBlur**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#GaussianBlur(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20double,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, double sigmaX, double sigmaY) Blurs an image using a Gaussian filter. |
| static void | [**GaussianBlur**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#GaussianBlur(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20double,%20double,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, double sigmaX, double sigmaY, int borderType) Blurs an image using a Gaussian filter. |
| static [Mat](http://docs.google.com/org/opencv/core/Mat.html) | [**getAffineTransform**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getAffineTransform(org.opencv.core.MatOfPoint2f,%20org.opencv.core.MatOfPoint2f))([MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) src, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) dst) |
| static [Mat](http://docs.google.com/org/opencv/core/Mat.html) | [**getDefaultNewCameraMatrix**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getDefaultNewCameraMatrix(org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix) Returns the default new camera matrix. |
| static [Mat](http://docs.google.com/org/opencv/core/Mat.html) | [**getDefaultNewCameraMatrix**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getDefaultNewCameraMatrix(org.opencv.core.Mat,%20org.opencv.core.Size))([Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Size](http://docs.google.com/org/opencv/core/Size.html) imgsize) Returns the default new camera matrix. |
| static [Mat](http://docs.google.com/org/opencv/core/Mat.html) | [**getDefaultNewCameraMatrix**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getDefaultNewCameraMatrix(org.opencv.core.Mat,%20org.opencv.core.Size,%20boolean))([Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Size](http://docs.google.com/org/opencv/core/Size.html) imgsize, boolean centerPrincipalPoint) Returns the default new camera matrix. |
| static void | [**getDerivKernels**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getDerivKernels(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) kx, [Mat](http://docs.google.com/org/opencv/core/Mat.html) ky, int dx, int dy, int ksize) Returns filter coefficients for computing spatial image derivatives. |
| static void | [**getDerivKernels**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getDerivKernels(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20int,%20boolean))([Mat](http://docs.google.com/org/opencv/core/Mat.html) kx, [Mat](http://docs.google.com/org/opencv/core/Mat.html) ky, int dx, int dy, int ksize, boolean normalize) Returns filter coefficients for computing spatial image derivatives. |
| static void | [**getDerivKernels**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getDerivKernels(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20int,%20boolean,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) kx, [Mat](http://docs.google.com/org/opencv/core/Mat.html) ky, int dx, int dy, int ksize, boolean normalize, int ktype) Returns filter coefficients for computing spatial image derivatives. |
| static double | [**getFontScaleFromHeight**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getFontScaleFromHeight(int,%20int))(int fontFace, int pixelHeight) Calculates the font-specific size to use to achieve a given height in pixels. |
| static double | [**getFontScaleFromHeight**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getFontScaleFromHeight(int,%20int,%20int))(int fontFace, int pixelHeight, int thickness) Calculates the font-specific size to use to achieve a given height in pixels. |
| static [Mat](http://docs.google.com/org/opencv/core/Mat.html) | [**getGaborKernel**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getGaborKernel(org.opencv.core.Size,%20double,%20double,%20double,%20double))([Size](http://docs.google.com/org/opencv/core/Size.html) ksize, double sigma, double theta, double lambd, double gamma) Returns Gabor filter coefficients. |
| static [Mat](http://docs.google.com/org/opencv/core/Mat.html) | [**getGaborKernel**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getGaborKernel(org.opencv.core.Size,%20double,%20double,%20double,%20double,%20double))([Size](http://docs.google.com/org/opencv/core/Size.html) ksize, double sigma, double theta, double lambd, double gamma, double psi) Returns Gabor filter coefficients. |
| static [Mat](http://docs.google.com/org/opencv/core/Mat.html) | [**getGaborKernel**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getGaborKernel(org.opencv.core.Size,%20double,%20double,%20double,%20double,%20double,%20int))([Size](http://docs.google.com/org/opencv/core/Size.html) ksize, double sigma, double theta, double lambd, double gamma, double psi, int ktype) Returns Gabor filter coefficients. |
| static [Mat](http://docs.google.com/org/opencv/core/Mat.html) | [**getGaussianKernel**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getGaussianKernel(int,%20double))(int ksize, double sigma) Returns Gaussian filter coefficients. |
| static [Mat](http://docs.google.com/org/opencv/core/Mat.html) | [**getGaussianKernel**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getGaussianKernel(int,%20double,%20int))(int ksize, double sigma, int ktype) Returns Gaussian filter coefficients. |
| static [Mat](http://docs.google.com/org/opencv/core/Mat.html) | [**getPerspectiveTransform**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getPerspectiveTransform(org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Calculates a perspective transform from four pairs of the corresponding points. |
| static void | [**getRectSubPix**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getRectSubPix(org.opencv.core.Mat,%20org.opencv.core.Size,%20org.opencv.core.Point,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Size](http://docs.google.com/org/opencv/core/Size.html) patchSize, [Point](http://docs.google.com/org/opencv/core/Point.html) center, [Mat](http://docs.google.com/org/opencv/core/Mat.html) patch) Retrieves a pixel rectangle from an image with sub-pixel accuracy. |
| static void | [**getRectSubPix**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getRectSubPix(org.opencv.core.Mat,%20org.opencv.core.Size,%20org.opencv.core.Point,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Size](http://docs.google.com/org/opencv/core/Size.html) patchSize, [Point](http://docs.google.com/org/opencv/core/Point.html) center, [Mat](http://docs.google.com/org/opencv/core/Mat.html) patch, int patchType) Retrieves a pixel rectangle from an image with sub-pixel accuracy. |
| static [Mat](http://docs.google.com/org/opencv/core/Mat.html) | [**getRotationMatrix2D**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getRotationMatrix2D(org.opencv.core.Point,%20double,%20double))([Point](http://docs.google.com/org/opencv/core/Point.html) center, double angle, double scale) Calculates an affine matrix of 2D rotation. |
| static [Mat](http://docs.google.com/org/opencv/core/Mat.html) | [**getStructuringElement**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getStructuringElement(int,%20org.opencv.core.Size))(int shape, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize) Returns a structuring element of the specified size and shape for morphological operations. |
| static [Mat](http://docs.google.com/org/opencv/core/Mat.html) | [**getStructuringElement**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getStructuringElement(int,%20org.opencv.core.Size,%20org.opencv.core.Point))(int shape, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor) Returns a structuring element of the specified size and shape for morphological operations. |
| static [Size](http://docs.google.com/org/opencv/core/Size.html) | [**getTextSize**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#getTextSize(java.lang.String,%20int,%20double,%20int,%20int%5B%5D))(java.lang.String text, int fontFace, double fontScale, int thickness, int[] baseLine) |
| static void | [**goodFeaturesToTrack**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#goodFeaturesToTrack(org.opencv.core.Mat,%20org.opencv.core.MatOfPoint,%20int,%20double,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) corners, int maxCorners, double qualityLevel, double minDistance) Determines strong corners on an image. |
| static void | [**goodFeaturesToTrack**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#goodFeaturesToTrack(org.opencv.core.Mat,%20org.opencv.core.MatOfPoint,%20int,%20double,%20double,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) corners, int maxCorners, double qualityLevel, double minDistance, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask) Determines strong corners on an image. |
| static void | [**goodFeaturesToTrack**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#goodFeaturesToTrack(org.opencv.core.Mat,%20org.opencv.core.MatOfPoint,%20int,%20double,%20double,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) corners, int maxCorners, double qualityLevel, double minDistance, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, int blockSize) Determines strong corners on an image. |
| static void | [**goodFeaturesToTrack**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#goodFeaturesToTrack(org.opencv.core.Mat,%20org.opencv.core.MatOfPoint,%20int,%20double,%20double,%20org.opencv.core.Mat,%20int,%20boolean))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) corners, int maxCorners, double qualityLevel, double minDistance, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, int blockSize, boolean useHarrisDetector) Determines strong corners on an image. |
| static void | [**goodFeaturesToTrack**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#goodFeaturesToTrack(org.opencv.core.Mat,%20org.opencv.core.MatOfPoint,%20int,%20double,%20double,%20org.opencv.core.Mat,%20int,%20boolean,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) corners, int maxCorners, double qualityLevel, double minDistance, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, int blockSize, boolean useHarrisDetector, double k) Determines strong corners on an image. |
| static void | [**goodFeaturesToTrack**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#goodFeaturesToTrack(org.opencv.core.Mat,%20org.opencv.core.MatOfPoint,%20int,%20double,%20double,%20org.opencv.core.Mat,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) corners, int maxCorners, double qualityLevel, double minDistance, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, int blockSize, int gradientSize) |
| static void | [**goodFeaturesToTrack**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#goodFeaturesToTrack(org.opencv.core.Mat,%20org.opencv.core.MatOfPoint,%20int,%20double,%20double,%20org.opencv.core.Mat,%20int,%20int,%20boolean))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) corners, int maxCorners, double qualityLevel, double minDistance, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, int blockSize, int gradientSize, boolean useHarrisDetector) |
| static void | [**goodFeaturesToTrack**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#goodFeaturesToTrack(org.opencv.core.Mat,%20org.opencv.core.MatOfPoint,%20int,%20double,%20double,%20org.opencv.core.Mat,%20int,%20int,%20boolean,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) corners, int maxCorners, double qualityLevel, double minDistance, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, int blockSize, int gradientSize, boolean useHarrisDetector, double k) |
| static void | [**grabCut**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#grabCut(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Rect,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Rect](http://docs.google.com/org/opencv/core/Rect.html) rect, [Mat](http://docs.google.com/org/opencv/core/Mat.html) bgdModel, [Mat](http://docs.google.com/org/opencv/core/Mat.html) fgdModel, int iterCount) Runs the GrabCut algorithm. |
| static void | [**grabCut**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#grabCut(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Rect,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Rect](http://docs.google.com/org/opencv/core/Rect.html) rect, [Mat](http://docs.google.com/org/opencv/core/Mat.html) bgdModel, [Mat](http://docs.google.com/org/opencv/core/Mat.html) fgdModel, int iterCount, int mode) Runs the GrabCut algorithm. |
| static void | [**HoughCircles**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HoughCircles(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20double,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) circles, int method, double dp, double minDist) Finds circles in a grayscale image using the Hough transform. |
| static void | [**HoughCircles**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HoughCircles(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20double,%20double,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) circles, int method, double dp, double minDist, double param1) Finds circles in a grayscale image using the Hough transform. |
| static void | [**HoughCircles**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HoughCircles(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20double,%20double,%20double,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) circles, int method, double dp, double minDist, double param1, double param2) Finds circles in a grayscale image using the Hough transform. |
| static void | [**HoughCircles**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HoughCircles(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20double,%20double,%20double,%20double,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) circles, int method, double dp, double minDist, double param1, double param2, int minRadius) Finds circles in a grayscale image using the Hough transform. |
| static void | [**HoughCircles**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HoughCircles(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20double,%20double,%20double,%20double,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) circles, int method, double dp, double minDist, double param1, double param2, int minRadius, int maxRadius) Finds circles in a grayscale image using the Hough transform. |
| static void | [**HoughLines**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HoughLines(org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20double,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines, double rho, double theta, int threshold) Finds lines in a binary image using the standard Hough transform. |
| static void | [**HoughLines**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HoughLines(org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20double,%20int,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines, double rho, double theta, int threshold, double srn) Finds lines in a binary image using the standard Hough transform. |
| static void | [**HoughLines**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HoughLines(org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20double,%20int,%20double,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines, double rho, double theta, int threshold, double srn, double stn) Finds lines in a binary image using the standard Hough transform. |
| static void | [**HoughLines**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HoughLines(org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20double,%20int,%20double,%20double,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines, double rho, double theta, int threshold, double srn, double stn, double min\_theta) Finds lines in a binary image using the standard Hough transform. |
| static void | [**HoughLines**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HoughLines(org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20double,%20int,%20double,%20double,%20double,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines, double rho, double theta, int threshold, double srn, double stn, double min\_theta, double max\_theta) Finds lines in a binary image using the standard Hough transform. |
| static void | [**HoughLinesP**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HoughLinesP(org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20double,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines, double rho, double theta, int threshold) Finds line segments in a binary image using the probabilistic Hough transform. |
| static void | [**HoughLinesP**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HoughLinesP(org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20double,%20int,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines, double rho, double theta, int threshold, double minLineLength) Finds line segments in a binary image using the probabilistic Hough transform. |
| static void | [**HoughLinesP**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HoughLinesP(org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20double,%20int,%20double,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines, double rho, double theta, int threshold, double minLineLength, double maxLineGap) Finds line segments in a binary image using the probabilistic Hough transform. |
| static void | [**HoughLinesPointSet**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HoughLinesPointSet(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20double,%20double,%20double,%20double,%20double,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) \_point, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_lines, int lines\_max, int threshold, double min\_rho, double max\_rho, double rho\_step, double min\_theta, double max\_theta, double theta\_step) Finds lines in a set of points using the standard Hough transform. |
| static void | [**HuMoments**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#HuMoments(org.opencv.imgproc.Moments,%20org.opencv.core.Mat))([Moments](http://docs.google.com/org/opencv/imgproc/Moments.html) m, [Mat](http://docs.google.com/org/opencv/core/Mat.html) hu) |
| static void | [**initUndistortRectifyMap**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#initUndistortRectifyMap(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20int,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) R, [Mat](http://docs.google.com/org/opencv/core/Mat.html) newCameraMatrix, [Size](http://docs.google.com/org/opencv/core/Size.html) size, int m1type, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map2) Computes the undistortion and rectification transformation map. |
| static float | [**initWideAngleProjMap**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#initWideAngleProjMap(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20int,%20int,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs, [Size](http://docs.google.com/org/opencv/core/Size.html) imageSize, int destImageWidth, int m1type, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map2) |
| static float | [**initWideAngleProjMap**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#initWideAngleProjMap(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20int,%20int,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs, [Size](http://docs.google.com/org/opencv/core/Size.html) imageSize, int destImageWidth, int m1type, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map2, int projType) |
| static float | [**initWideAngleProjMap**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#initWideAngleProjMap(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20int,%20int,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs, [Size](http://docs.google.com/org/opencv/core/Size.html) imageSize, int destImageWidth, int m1type, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map2, int projType, double alpha) |
| static void | [**integral**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#integral(org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sum) |
| static void | [**integral**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#integral(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sum, int sdepth) |
| static void | [**integral2**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#integral2(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sum, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sqsum) |
| static void | [**integral2**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#integral2(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sum, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sqsum, int sdepth) |
| static void | [**integral2**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#integral2(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sum, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sqsum, int sdepth, int sqdepth) |
| static void | [**integral3**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#integral3(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sum, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sqsum, [Mat](http://docs.google.com/org/opencv/core/Mat.html) tilted) Calculates the integral of an image. |
| static void | [**integral3**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#integral3(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sum, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sqsum, [Mat](http://docs.google.com/org/opencv/core/Mat.html) tilted, int sdepth) Calculates the integral of an image. |
| static void | [**integral3**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#integral3(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sum, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sqsum, [Mat](http://docs.google.com/org/opencv/core/Mat.html) tilted, int sdepth, int sqdepth) Calculates the integral of an image. |
| static float | [**intersectConvexConvex**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#intersectConvexConvex(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) \_p1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_p2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_p12) Finds intersection of two convex polygons |
| static float | [**intersectConvexConvex**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#intersectConvexConvex(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20boolean))([Mat](http://docs.google.com/org/opencv/core/Mat.html) \_p1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_p2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_p12, boolean handleNested) Finds intersection of two convex polygons |
| static void | [**invertAffineTransform**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#invertAffineTransform(org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) M, [Mat](http://docs.google.com/org/opencv/core/Mat.html) iM) Inverts an affine transformation. |
| static boolean | [**isContourConvex**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#isContourConvex(org.opencv.core.MatOfPoint))([MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) contour) Tests a contour convexity. |
| static void | [**Laplacian**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#Laplacian(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth) Calculates the Laplacian of an image. |
| static void | [**Laplacian**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#Laplacian(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int ksize) Calculates the Laplacian of an image. |
| static void | [**Laplacian**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#Laplacian(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int ksize, double scale) Calculates the Laplacian of an image. |
| static void | [**Laplacian**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#Laplacian(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20double,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int ksize, double scale, double delta) Calculates the Laplacian of an image. |
| static void | [**Laplacian**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#Laplacian(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20double,%20double,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int ksize, double scale, double delta, int borderType) Calculates the Laplacian of an image. |
| static void | [**line**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#line(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Point,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Draws a line segment connecting two points. |
| static void | [**line**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#line(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Point,%20org.opencv.core.Scalar,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness) Draws a line segment connecting two points. |
| static void | [**line**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#line(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Point,%20org.opencv.core.Scalar,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType) Draws a line segment connecting two points. |
| static void | [**line**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#line(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Point,%20org.opencv.core.Scalar,%20int,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType, int shift) Draws a line segment connecting two points. |
| static double | [**matchShapes**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#matchShapes(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) contour1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) contour2, int method, double parameter) Compares two shapes. |
| static void | [**matchTemplate**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#matchTemplate(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) templ, [Mat](http://docs.google.com/org/opencv/core/Mat.html) result, int method) Compares a template against overlapped image regions. |
| static void | [**matchTemplate**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#matchTemplate(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) templ, [Mat](http://docs.google.com/org/opencv/core/Mat.html) result, int method, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask) Compares a template against overlapped image regions. |
| static void | [**medianBlur**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#medianBlur(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ksize) Blurs an image using the median filter. |
| static [RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) | [**minAreaRect**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#minAreaRect(org.opencv.core.MatOfPoint2f))([MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) points) Finds a rotated rectangle of the minimum area enclosing the input 2D point set. |
| static void | [**minEnclosingCircle**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#minEnclosingCircle(org.opencv.core.MatOfPoint2f,%20org.opencv.core.Point,%20float%5B%5D))([MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) points, [Point](http://docs.google.com/org/opencv/core/Point.html) center, float[] radius) Finds a circle of the minimum area enclosing a 2D point set. |
| static double | [**minEnclosingTriangle**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#minEnclosingTriangle(org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) points, [Mat](http://docs.google.com/org/opencv/core/Mat.html) triangle) Finds a triangle of minimum area enclosing a 2D point set and returns its area. |
| static [Moments](http://docs.google.com/org/opencv/imgproc/Moments.html) | [**moments**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#moments(org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) array) Calculates all of the moments up to the third order of a polygon or rasterized shape. |
| static [Moments](http://docs.google.com/org/opencv/imgproc/Moments.html) | [**moments**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#moments(org.opencv.core.Mat,%20boolean))([Mat](http://docs.google.com/org/opencv/core/Mat.html) array, boolean binaryImage) Calculates all of the moments up to the third order of a polygon or rasterized shape. |
| static void | [**morphologyEx**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#morphologyEx(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int op, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel) Performs advanced morphological transformations. |
| static void | [**morphologyEx**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#morphologyEx(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20org.opencv.core.Point))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int op, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor) Performs advanced morphological transformations. |
| static void | [**morphologyEx**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#morphologyEx(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20org.opencv.core.Point,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int op, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int iterations) Performs advanced morphological transformations. |
| static void | [**morphologyEx**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#morphologyEx(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20org.opencv.core.Point,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int op, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int iterations, int borderType) Performs advanced morphological transformations. |
| static void | [**morphologyEx**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#morphologyEx(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20org.opencv.core.Point,%20int,%20int,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int op, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int iterations, int borderType, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) borderValue) Performs advanced morphological transformations. |
| static [Point](http://docs.google.com/org/opencv/core/Point.html) | [**phaseCorrelate**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#phaseCorrelate(org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) src2) The function is used to detect translational shifts that occur between two images. |
| static [Point](http://docs.google.com/org/opencv/core/Point.html) | [**phaseCorrelate**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#phaseCorrelate(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) src2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) window) The function is used to detect translational shifts that occur between two images. |
| static [Point](http://docs.google.com/org/opencv/core/Point.html) | [**phaseCorrelate**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#phaseCorrelate(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20double%5B%5D))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) src2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) window, double[] response) The function is used to detect translational shifts that occur between two images. |
| static double | [**pointPolygonTest**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#pointPolygonTest(org.opencv.core.MatOfPoint2f,%20org.opencv.core.Point,%20boolean))([MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) contour, [Point](http://docs.google.com/org/opencv/core/Point.html) pt, boolean measureDist) Performs a point-in-contour test. |
| static void | [**polylines**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#polylines(org.opencv.core.Mat,%20java.util.List,%20boolean,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> pts, boolean isClosed, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Draws several polygonal curves. |
| static void | [**polylines**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#polylines(org.opencv.core.Mat,%20java.util.List,%20boolean,%20org.opencv.core.Scalar,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> pts, boolean isClosed, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness) Draws several polygonal curves. |
| static void | [**polylines**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#polylines(org.opencv.core.Mat,%20java.util.List,%20boolean,%20org.opencv.core.Scalar,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> pts, boolean isClosed, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType) Draws several polygonal curves. |
| static void | [**polylines**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#polylines(org.opencv.core.Mat,%20java.util.List,%20boolean,%20org.opencv.core.Scalar,%20int,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> pts, boolean isClosed, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType, int shift) Draws several polygonal curves. |
| static void | [**preCornerDetect**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#preCornerDetect(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ksize) Calculates a feature map for corner detection. |
| static void | [**preCornerDetect**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#preCornerDetect(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ksize, int borderType) Calculates a feature map for corner detection. |
| static void | [**putText**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#putText(org.opencv.core.Mat,%20java.lang.String,%20org.opencv.core.Point,%20int,%20double,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.lang.String text, [Point](http://docs.google.com/org/opencv/core/Point.html) org, int fontFace, double fontScale, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Draws a text string. |
| static void | [**putText**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#putText(org.opencv.core.Mat,%20java.lang.String,%20org.opencv.core.Point,%20int,%20double,%20org.opencv.core.Scalar,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.lang.String text, [Point](http://docs.google.com/org/opencv/core/Point.html) org, int fontFace, double fontScale, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness) Draws a text string. |
| static void | [**putText**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#putText(org.opencv.core.Mat,%20java.lang.String,%20org.opencv.core.Point,%20int,%20double,%20org.opencv.core.Scalar,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.lang.String text, [Point](http://docs.google.com/org/opencv/core/Point.html) org, int fontFace, double fontScale, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType) Draws a text string. |
| static void | [**putText**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#putText(org.opencv.core.Mat,%20java.lang.String,%20org.opencv.core.Point,%20int,%20double,%20org.opencv.core.Scalar,%20int,%20int,%20boolean))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.lang.String text, [Point](http://docs.google.com/org/opencv/core/Point.html) org, int fontFace, double fontScale, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType, boolean bottomLeftOrigin) Draws a text string. |
| static void | [**pyrDown**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#pyrDown(org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Blurs an image and downsamples it. |
| static void | [**pyrDown**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#pyrDown(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) dstsize) Blurs an image and downsamples it. |
| static void | [**pyrDown**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#pyrDown(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) dstsize, int borderType) Blurs an image and downsamples it. |
| static void | [**pyrMeanShiftFiltering**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#pyrMeanShiftFiltering(org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, double sp, double sr) Performs initial step of meanshift segmentation of an image. |
| static void | [**pyrMeanShiftFiltering**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#pyrMeanShiftFiltering(org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20double,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, double sp, double sr, int maxLevel) Performs initial step of meanshift segmentation of an image. |
| static void | [**pyrMeanShiftFiltering**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#pyrMeanShiftFiltering(org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20double,%20int,%20org.opencv.core.TermCriteria))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, double sp, double sr, int maxLevel, [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) termcrit) Performs initial step of meanshift segmentation of an image. |
| static void | [**pyrUp**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#pyrUp(org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Upsamples an image and then blurs it. |
| static void | [**pyrUp**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#pyrUp(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) dstsize) Upsamples an image and then blurs it. |
| static void | [**pyrUp**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#pyrUp(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) dstsize, int borderType) Upsamples an image and then blurs it. |
| static void | [**rectangle**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#rectangle(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Point,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Draws a simple, thick, or filled up-right rectangle. |
| static void | [**rectangle**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#rectangle(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Point,%20org.opencv.core.Scalar,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness) Draws a simple, thick, or filled up-right rectangle. |
| static void | [**rectangle**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#rectangle(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Point,%20org.opencv.core.Scalar,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType) Draws a simple, thick, or filled up-right rectangle. |
| static void | [**rectangle**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#rectangle(org.opencv.core.Mat,%20org.opencv.core.Point,%20org.opencv.core.Point,%20org.opencv.core.Scalar,%20int,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType, int shift) Draws a simple, thick, or filled up-right rectangle. |
| static void | [**remap**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#remap(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map2, int interpolation) Applies a generic geometrical transformation to an image. |
| static void | [**remap**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#remap(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map2, int interpolation, int borderMode) Applies a generic geometrical transformation to an image. |
| static void | [**remap**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#remap(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map2, int interpolation, int borderMode, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) borderValue) Applies a generic geometrical transformation to an image. |
| static void | [**resize**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#resize(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize) Resizes an image. |
| static void | [**resize**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#resize(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, double fx) Resizes an image. |
| static void | [**resize**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#resize(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20double,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, double fx, double fy) Resizes an image. |
| static void | [**resize**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#resize(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20double,%20double,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, double fx, double fy, int interpolation) Resizes an image. |
| static int | [**rotatedRectangleIntersection**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#rotatedRectangleIntersection(org.opencv.core.RotatedRect,%20org.opencv.core.RotatedRect,%20org.opencv.core.Mat))([RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) rect1, [RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) rect2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) intersectingRegion) Finds out if there is any intersection between two rotated rectangles. |
| static void | [**Scharr**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#Scharr(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int dx, int dy) Calculates the first x- or y- image derivative using Scharr operator. |
| static void | [**Scharr**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#Scharr(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20int,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int dx, int dy, double scale) Calculates the first x- or y- image derivative using Scharr operator. |
| static void | [**Scharr**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#Scharr(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20int,%20double,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int dx, int dy, double scale, double delta) Calculates the first x- or y- image derivative using Scharr operator. |
| static void | [**Scharr**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#Scharr(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20int,%20double,%20double,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int dx, int dy, double scale, double delta, int borderType) Calculates the first x- or y- image derivative using Scharr operator. |
| static void | [**sepFilter2D**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#sepFilter2D(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernelX, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernelY) Applies a separable linear filter to an image. |
| static void | [**sepFilter2D**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#sepFilter2D(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Point))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernelX, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernelY, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor) Applies a separable linear filter to an image. |
| static void | [**sepFilter2D**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#sepFilter2D(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Point,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernelX, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernelY, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, double delta) Applies a separable linear filter to an image. |
| static void | [**sepFilter2D**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#sepFilter2D(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Point,%20double,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernelX, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernelY, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, double delta, int borderType) Applies a separable linear filter to an image. |
| static void | [**Sobel**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#Sobel(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int dx, int dy) Calculates the first, second, third, or mixed image derivatives using an extended Sobel operator. |
| static void | [**Sobel**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#Sobel(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int dx, int dy, int ksize) Calculates the first, second, third, or mixed image derivatives using an extended Sobel operator. |
| static void | [**Sobel**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#Sobel(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20int,%20int,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int dx, int dy, int ksize, double scale) Calculates the first, second, third, or mixed image derivatives using an extended Sobel operator. |
| static void | [**Sobel**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#Sobel(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20int,%20int,%20double,%20double))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int dx, int dy, int ksize, double scale, double delta) Calculates the first, second, third, or mixed image derivatives using an extended Sobel operator. |
| static void | [**Sobel**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#Sobel(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int,%20int,%20int,%20double,%20double,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int dx, int dy, int ksize, double scale, double delta, int borderType) Calculates the first, second, third, or mixed image derivatives using an extended Sobel operator. |
| static void | [**spatialGradient**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#spatialGradient(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dx, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dy) Calculates the first order image derivative in both x and y using a Sobel operator Equivalent to calling: Sobel( src, dx, CV\_16SC1, 1, 0, 3 ); Sobel( src, dy, CV\_16SC1, 0, 1, 3 ); |
| static void | [**spatialGradient**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#spatialGradient(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dx, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dy, int ksize) Calculates the first order image derivative in both x and y using a Sobel operator Equivalent to calling: Sobel( src, dx, CV\_16SC1, 1, 0, 3 ); Sobel( src, dy, CV\_16SC1, 0, 1, 3 ); |
| static void | [**spatialGradient**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#spatialGradient(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dx, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dy, int ksize, int borderType) Calculates the first order image derivative in both x and y using a Sobel operator Equivalent to calling: Sobel( src, dx, CV\_16SC1, 1, 0, 3 ); Sobel( src, dy, CV\_16SC1, 0, 1, 3 ); |
| static void | [**sqrBoxFilter**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#sqrBoxFilter(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Size))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize) Calculates the normalized sum of squares of the pixel values overlapping the filter. |
| static void | [**sqrBoxFilter**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#sqrBoxFilter(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Size,%20org.opencv.core.Point))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor) Calculates the normalized sum of squares of the pixel values overlapping the filter. |
| static void | [**sqrBoxFilter**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#sqrBoxFilter(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Size,%20org.opencv.core.Point,%20boolean))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, boolean normalize) Calculates the normalized sum of squares of the pixel values overlapping the filter. |
| static void | [**sqrBoxFilter**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#sqrBoxFilter(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20org.opencv.core.Size,%20org.opencv.core.Point,%20boolean,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, boolean normalize, int borderType) Calculates the normalized sum of squares of the pixel values overlapping the filter. |
| static double | [**threshold**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#threshold(org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20double,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, double thresh, double maxval, int type) Applies a fixed-level threshold to each array element. |
| static void | [**undistort**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#undistort(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs) Transforms an image to compensate for lens distortion. |
| static void | [**undistort**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#undistort(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) newCameraMatrix) Transforms an image to compensate for lens distortion. |
| static void | [**undistortPoints**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#undistortPoints(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs) Computes the ideal point coordinates from the observed point coordinates. |
| static void | [**undistortPoints**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#undistortPoints(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) R) Computes the ideal point coordinates from the observed point coordinates. |
| static void | [**undistortPoints**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#undistortPoints(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) R, [Mat](http://docs.google.com/org/opencv/core/Mat.html) P) Computes the ideal point coordinates from the observed point coordinates. |
| static void | [**undistortPointsIter**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#undistortPointsIter(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.TermCriteria))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) R, [Mat](http://docs.google.com/org/opencv/core/Mat.html) P, [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) criteria) **Note:** Default version of #undistortPoints does 5 iterations to compute undistorted points. |
| static void | [**warpAffine**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#warpAffine(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) M, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize) Applies an affine transformation to an image. |
| static void | [**warpAffine**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#warpAffine(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) M, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, int flags) Applies an affine transformation to an image. |
| static void | [**warpAffine**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#warpAffine(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) M, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, int flags, int borderMode) Applies an affine transformation to an image. |
| static void | [**warpAffine**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#warpAffine(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20int,%20int,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) M, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, int flags, int borderMode, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) borderValue) Applies an affine transformation to an image. |
| static void | [**warpPerspective**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#warpPerspective(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) M, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize) Applies a perspective transformation to an image. |
| static void | [**warpPerspective**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#warpPerspective(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) M, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, int flags) Applies a perspective transformation to an image. |
| static void | [**warpPerspective**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#warpPerspective(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) M, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, int flags, int borderMode) Applies a perspective transformation to an image. |
| static void | [**warpPerspective**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#warpPerspective(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20int,%20int,%20org.opencv.core.Scalar))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) M, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, int flags, int borderMode, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) borderValue) Applies a perspective transformation to an image. |
| static void | [**warpPolar**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#warpPolar(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Size,%20org.opencv.core.Point,%20double,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, [Point](http://docs.google.com/org/opencv/core/Point.html) center, double maxRadius, int flags) Remaps an image to polar or semilog-polar coordinates space polar\_remaps\_reference\_image ![Polar remaps reference](pics/polar\_remap\_doc.png) Transform the source image using the following transformation: \( dst(\rho , \phi ) = src(x,y) \) where \( \begin{array}{l} \vec{I} = (x - center.x, \;y - center.y) \\ \phi = Kangle \cdot \texttt{angle} (\vec{I}) \\ \rho = \left\{\begin{matrix} Klin \cdot \texttt{magnitude} (\vec{I}) & default \\ Klog \cdot log\_e(\texttt{magnitude} (\vec{I})) & if \; semilog \\ \end{matrix}\right. |
| static void | [**watershed**](http://docs.google.com/org/opencv/imgproc/Imgproc.html#watershed(org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) markers) Performs a marker-based image segmentation using the watershed algorithm. |

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

### Field Detail

#### ADAPTIVE\_THRESH\_GAUSSIAN\_C public static final int ADAPTIVE\_THRESH\_GAUSSIAN\_CSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.ADAPTIVE_THRESH_GAUSSIAN_C)

#### ADAPTIVE\_THRESH\_MEAN\_C public static final int ADAPTIVE\_THRESH\_MEAN\_CSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.ADAPTIVE_THRESH_MEAN_C)

#### CC\_STAT\_AREA public static final int CC\_STAT\_AREASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CC_STAT_AREA)

#### CC\_STAT\_HEIGHT public static final int CC\_STAT\_HEIGHTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CC_STAT_HEIGHT)

#### CC\_STAT\_LEFT public static final int CC\_STAT\_LEFTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CC_STAT_LEFT)

#### CC\_STAT\_MAX public static final int CC\_STAT\_MAXSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CC_STAT_MAX)

#### CC\_STAT\_TOP public static final int CC\_STAT\_TOPSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CC_STAT_TOP)

#### CC\_STAT\_WIDTH public static final int CC\_STAT\_WIDTHSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CC_STAT_WIDTH)

#### CCL\_BBDT public static final int CCL\_BBDTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CCL_BBDT)

#### CCL\_BOLELLI public static final int CCL\_BOLELLISee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CCL_BOLELLI)

#### CCL\_DEFAULT public static final int CCL\_DEFAULTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CCL_DEFAULT)

#### CCL\_GRANA public static final int CCL\_GRANASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CCL_GRANA)

#### CCL\_SAUF public static final int CCL\_SAUFSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CCL_SAUF)

#### CCL\_SPAGHETTI public static final int CCL\_SPAGHETTISee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CCL_SPAGHETTI)

#### CCL\_WU public static final int CCL\_WUSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CCL_WU)

#### CHAIN\_APPROX\_NONE public static final int CHAIN\_APPROX\_NONESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CHAIN_APPROX_NONE)

#### CHAIN\_APPROX\_SIMPLE public static final int CHAIN\_APPROX\_SIMPLESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CHAIN_APPROX_SIMPLE)

#### CHAIN\_APPROX\_TC89\_KCOS public static final int CHAIN\_APPROX\_TC89\_KCOSSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CHAIN_APPROX_TC89_KCOS)

#### CHAIN\_APPROX\_TC89\_L1 public static final int CHAIN\_APPROX\_TC89\_L1See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CHAIN_APPROX_TC89_L1)

#### COLOR\_BayerBG2BGR public static final int COLOR\_BayerBG2BGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerBG2BGR)

#### COLOR\_BayerBG2BGR\_EA public static final int COLOR\_BayerBG2BGR\_EASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerBG2BGR_EA)

#### COLOR\_BayerBG2BGR\_VNG public static final int COLOR\_BayerBG2BGR\_VNGSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerBG2BGR_VNG)

#### COLOR\_BayerBG2BGRA public static final int COLOR\_BayerBG2BGRASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerBG2BGRA)

#### COLOR\_BayerBG2GRAY public static final int COLOR\_BayerBG2GRAYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerBG2GRAY)

#### COLOR\_BayerBG2RGB public static final int COLOR\_BayerBG2RGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerBG2RGB)

#### COLOR\_BayerBG2RGB\_EA public static final int COLOR\_BayerBG2RGB\_EASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerBG2RGB_EA)

#### COLOR\_BayerBG2RGB\_VNG public static final int COLOR\_BayerBG2RGB\_VNGSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerBG2RGB_VNG)

#### COLOR\_BayerBG2RGBA public static final int COLOR\_BayerBG2RGBASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerBG2RGBA)

#### COLOR\_BayerGB2BGR public static final int COLOR\_BayerGB2BGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerGB2BGR)

#### COLOR\_BayerGB2BGR\_EA public static final int COLOR\_BayerGB2BGR\_EASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerGB2BGR_EA)

#### COLOR\_BayerGB2BGR\_VNG public static final int COLOR\_BayerGB2BGR\_VNGSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerGB2BGR_VNG)

#### COLOR\_BayerGB2BGRA public static final int COLOR\_BayerGB2BGRASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerGB2BGRA)

#### COLOR\_BayerGB2GRAY public static final int COLOR\_BayerGB2GRAYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerGB2GRAY)

#### COLOR\_BayerGB2RGB public static final int COLOR\_BayerGB2RGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerGB2RGB)

#### COLOR\_BayerGB2RGB\_EA public static final int COLOR\_BayerGB2RGB\_EASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerGB2RGB_EA)

#### COLOR\_BayerGB2RGB\_VNG public static final int COLOR\_BayerGB2RGB\_VNGSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerGB2RGB_VNG)

#### COLOR\_BayerGB2RGBA public static final int COLOR\_BayerGB2RGBASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerGB2RGBA)

#### COLOR\_BayerGR2BGR public static final int COLOR\_BayerGR2BGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerGR2BGR)

#### COLOR\_BayerGR2BGR\_EA public static final int COLOR\_BayerGR2BGR\_EASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerGR2BGR_EA)

#### COLOR\_BayerGR2BGR\_VNG public static final int COLOR\_BayerGR2BGR\_VNGSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerGR2BGR_VNG)

#### COLOR\_BayerGR2BGRA public static final int COLOR\_BayerGR2BGRASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerGR2BGRA)

#### COLOR\_BayerGR2GRAY public static final int COLOR\_BayerGR2GRAYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerGR2GRAY)

#### COLOR\_BayerGR2RGB public static final int COLOR\_BayerGR2RGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerGR2RGB)

#### COLOR\_BayerGR2RGB\_EA public static final int COLOR\_BayerGR2RGB\_EASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerGR2RGB_EA)

#### COLOR\_BayerGR2RGB\_VNG public static final int COLOR\_BayerGR2RGB\_VNGSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerGR2RGB_VNG)

#### COLOR\_BayerGR2RGBA public static final int COLOR\_BayerGR2RGBASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerGR2RGBA)

#### COLOR\_BayerRG2BGR public static final int COLOR\_BayerRG2BGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerRG2BGR)

#### COLOR\_BayerRG2BGR\_EA public static final int COLOR\_BayerRG2BGR\_EASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerRG2BGR_EA)

#### COLOR\_BayerRG2BGR\_VNG public static final int COLOR\_BayerRG2BGR\_VNGSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerRG2BGR_VNG)

#### COLOR\_BayerRG2BGRA public static final int COLOR\_BayerRG2BGRASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerRG2BGRA)

#### COLOR\_BayerRG2GRAY public static final int COLOR\_BayerRG2GRAYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerRG2GRAY)

#### COLOR\_BayerRG2RGB public static final int COLOR\_BayerRG2RGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerRG2RGB)

#### COLOR\_BayerRG2RGB\_EA public static final int COLOR\_BayerRG2RGB\_EASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerRG2RGB_EA)

#### COLOR\_BayerRG2RGB\_VNG public static final int COLOR\_BayerRG2RGB\_VNGSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerRG2RGB_VNG)

#### COLOR\_BayerRG2RGBA public static final int COLOR\_BayerRG2RGBASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BayerRG2RGBA)

#### COLOR\_BGR2BGR555 public static final int COLOR\_BGR2BGR555See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR2BGR555)

#### COLOR\_BGR2BGR565 public static final int COLOR\_BGR2BGR565See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR2BGR565)

#### COLOR\_BGR2BGRA public static final int COLOR\_BGR2BGRASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR2BGRA)

#### COLOR\_BGR2GRAY public static final int COLOR\_BGR2GRAYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR2GRAY)

#### COLOR\_BGR2HLS public static final int COLOR\_BGR2HLSSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR2HLS)

#### COLOR\_BGR2HLS\_FULL public static final int COLOR\_BGR2HLS\_FULLSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR2HLS_FULL)

#### COLOR\_BGR2HSV public static final int COLOR\_BGR2HSVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR2HSV)

#### COLOR\_BGR2HSV\_FULL public static final int COLOR\_BGR2HSV\_FULLSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR2HSV_FULL)

#### COLOR\_BGR2Lab public static final int COLOR\_BGR2LabSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR2Lab)

#### COLOR\_BGR2Luv public static final int COLOR\_BGR2LuvSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR2Luv)

#### COLOR\_BGR2RGB public static final int COLOR\_BGR2RGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR2RGB)

#### COLOR\_BGR2RGBA public static final int COLOR\_BGR2RGBASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR2RGBA)

#### COLOR\_BGR2XYZ public static final int COLOR\_BGR2XYZSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR2XYZ)

#### COLOR\_BGR2YCrCb public static final int COLOR\_BGR2YCrCbSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR2YCrCb)

#### COLOR\_BGR2YUV public static final int COLOR\_BGR2YUVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR2YUV)

#### COLOR\_BGR2YUV\_I420 public static final int COLOR\_BGR2YUV\_I420See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR2YUV_I420)

#### COLOR\_BGR2YUV\_IYUV public static final int COLOR\_BGR2YUV\_IYUVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR2YUV_IYUV)

#### COLOR\_BGR2YUV\_YV12 public static final int COLOR\_BGR2YUV\_YV12See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR2YUV_YV12)

#### COLOR\_BGR5552BGR public static final int COLOR\_BGR5552BGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR5552BGR)

#### COLOR\_BGR5552BGRA public static final int COLOR\_BGR5552BGRASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR5552BGRA)

#### COLOR\_BGR5552GRAY public static final int COLOR\_BGR5552GRAYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR5552GRAY)

#### COLOR\_BGR5552RGB public static final int COLOR\_BGR5552RGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR5552RGB)

#### COLOR\_BGR5552RGBA public static final int COLOR\_BGR5552RGBASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR5552RGBA)

#### COLOR\_BGR5652BGR public static final int COLOR\_BGR5652BGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR5652BGR)

#### COLOR\_BGR5652BGRA public static final int COLOR\_BGR5652BGRASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR5652BGRA)

#### COLOR\_BGR5652GRAY public static final int COLOR\_BGR5652GRAYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR5652GRAY)

#### COLOR\_BGR5652RGB public static final int COLOR\_BGR5652RGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR5652RGB)

#### COLOR\_BGR5652RGBA public static final int COLOR\_BGR5652RGBASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGR5652RGBA)

#### COLOR\_BGRA2BGR public static final int COLOR\_BGRA2BGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGRA2BGR)

#### COLOR\_BGRA2BGR555 public static final int COLOR\_BGRA2BGR555See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGRA2BGR555)

#### COLOR\_BGRA2BGR565 public static final int COLOR\_BGRA2BGR565See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGRA2BGR565)

#### COLOR\_BGRA2GRAY public static final int COLOR\_BGRA2GRAYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGRA2GRAY)

#### COLOR\_BGRA2RGB public static final int COLOR\_BGRA2RGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGRA2RGB)

#### COLOR\_BGRA2RGBA public static final int COLOR\_BGRA2RGBASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGRA2RGBA)

#### COLOR\_BGRA2YUV\_I420 public static final int COLOR\_BGRA2YUV\_I420See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGRA2YUV_I420)

#### COLOR\_BGRA2YUV\_IYUV public static final int COLOR\_BGRA2YUV\_IYUVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGRA2YUV_IYUV)

#### COLOR\_BGRA2YUV\_YV12 public static final int COLOR\_BGRA2YUV\_YV12See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_BGRA2YUV_YV12)

#### COLOR\_COLORCVT\_MAX public static final int COLOR\_COLORCVT\_MAXSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_COLORCVT_MAX)

#### COLOR\_GRAY2BGR public static final int COLOR\_GRAY2BGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_GRAY2BGR)

#### COLOR\_GRAY2BGR555 public static final int COLOR\_GRAY2BGR555See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_GRAY2BGR555)

#### COLOR\_GRAY2BGR565 public static final int COLOR\_GRAY2BGR565See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_GRAY2BGR565)

#### COLOR\_GRAY2BGRA public static final int COLOR\_GRAY2BGRASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_GRAY2BGRA)

#### COLOR\_GRAY2RGB public static final int COLOR\_GRAY2RGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_GRAY2RGB)

#### COLOR\_GRAY2RGBA public static final int COLOR\_GRAY2RGBASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_GRAY2RGBA)

#### COLOR\_HLS2BGR public static final int COLOR\_HLS2BGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_HLS2BGR)

#### COLOR\_HLS2BGR\_FULL public static final int COLOR\_HLS2BGR\_FULLSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_HLS2BGR_FULL)

#### COLOR\_HLS2RGB public static final int COLOR\_HLS2RGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_HLS2RGB)

#### COLOR\_HLS2RGB\_FULL public static final int COLOR\_HLS2RGB\_FULLSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_HLS2RGB_FULL)

#### COLOR\_HSV2BGR public static final int COLOR\_HSV2BGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_HSV2BGR)

#### COLOR\_HSV2BGR\_FULL public static final int COLOR\_HSV2BGR\_FULLSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_HSV2BGR_FULL)

#### COLOR\_HSV2RGB public static final int COLOR\_HSV2RGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_HSV2RGB)

#### COLOR\_HSV2RGB\_FULL public static final int COLOR\_HSV2RGB\_FULLSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_HSV2RGB_FULL)

#### COLOR\_Lab2BGR public static final int COLOR\_Lab2BGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_Lab2BGR)

#### COLOR\_Lab2LBGR public static final int COLOR\_Lab2LBGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_Lab2LBGR)

#### COLOR\_Lab2LRGB public static final int COLOR\_Lab2LRGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_Lab2LRGB)

#### COLOR\_Lab2RGB public static final int COLOR\_Lab2RGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_Lab2RGB)

#### COLOR\_LBGR2Lab public static final int COLOR\_LBGR2LabSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_LBGR2Lab)

#### COLOR\_LBGR2Luv public static final int COLOR\_LBGR2LuvSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_LBGR2Luv)

#### COLOR\_LRGB2Lab public static final int COLOR\_LRGB2LabSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_LRGB2Lab)

#### COLOR\_LRGB2Luv public static final int COLOR\_LRGB2LuvSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_LRGB2Luv)

#### COLOR\_Luv2BGR public static final int COLOR\_Luv2BGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_Luv2BGR)

#### COLOR\_Luv2LBGR public static final int COLOR\_Luv2LBGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_Luv2LBGR)

#### COLOR\_Luv2LRGB public static final int COLOR\_Luv2LRGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_Luv2LRGB)

#### COLOR\_Luv2RGB public static final int COLOR\_Luv2RGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_Luv2RGB)

#### COLOR\_mRGBA2RGBA public static final int COLOR\_mRGBA2RGBASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_mRGBA2RGBA)

#### COLOR\_RGB2BGR public static final int COLOR\_RGB2BGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGB2BGR)

#### COLOR\_RGB2BGR555 public static final int COLOR\_RGB2BGR555See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGB2BGR555)

#### COLOR\_RGB2BGR565 public static final int COLOR\_RGB2BGR565See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGB2BGR565)

#### COLOR\_RGB2BGRA public static final int COLOR\_RGB2BGRASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGB2BGRA)

#### COLOR\_RGB2GRAY public static final int COLOR\_RGB2GRAYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGB2GRAY)

#### COLOR\_RGB2HLS public static final int COLOR\_RGB2HLSSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGB2HLS)

#### COLOR\_RGB2HLS\_FULL public static final int COLOR\_RGB2HLS\_FULLSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGB2HLS_FULL)

#### COLOR\_RGB2HSV public static final int COLOR\_RGB2HSVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGB2HSV)

#### COLOR\_RGB2HSV\_FULL public static final int COLOR\_RGB2HSV\_FULLSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGB2HSV_FULL)

#### COLOR\_RGB2Lab public static final int COLOR\_RGB2LabSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGB2Lab)

#### COLOR\_RGB2Luv public static final int COLOR\_RGB2LuvSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGB2Luv)

#### COLOR\_RGB2RGBA public static final int COLOR\_RGB2RGBASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGB2RGBA)

#### COLOR\_RGB2XYZ public static final int COLOR\_RGB2XYZSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGB2XYZ)

#### COLOR\_RGB2YCrCb public static final int COLOR\_RGB2YCrCbSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGB2YCrCb)

#### COLOR\_RGB2YUV public static final int COLOR\_RGB2YUVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGB2YUV)

#### COLOR\_RGB2YUV\_I420 public static final int COLOR\_RGB2YUV\_I420See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGB2YUV_I420)

#### COLOR\_RGB2YUV\_IYUV public static final int COLOR\_RGB2YUV\_IYUVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGB2YUV_IYUV)

#### COLOR\_RGB2YUV\_YV12 public static final int COLOR\_RGB2YUV\_YV12See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGB2YUV_YV12)

#### COLOR\_RGBA2BGR public static final int COLOR\_RGBA2BGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGBA2BGR)

#### COLOR\_RGBA2BGR555 public static final int COLOR\_RGBA2BGR555See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGBA2BGR555)

#### COLOR\_RGBA2BGR565 public static final int COLOR\_RGBA2BGR565See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGBA2BGR565)

#### COLOR\_RGBA2BGRA public static final int COLOR\_RGBA2BGRASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGBA2BGRA)

#### COLOR\_RGBA2GRAY public static final int COLOR\_RGBA2GRAYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGBA2GRAY)

#### COLOR\_RGBA2mRGBA public static final int COLOR\_RGBA2mRGBASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGBA2mRGBA)

#### COLOR\_RGBA2RGB public static final int COLOR\_RGBA2RGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGBA2RGB)

#### COLOR\_RGBA2YUV\_I420 public static final int COLOR\_RGBA2YUV\_I420See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGBA2YUV_I420)

#### COLOR\_RGBA2YUV\_IYUV public static final int COLOR\_RGBA2YUV\_IYUVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGBA2YUV_IYUV)

#### COLOR\_RGBA2YUV\_YV12 public static final int COLOR\_RGBA2YUV\_YV12See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_RGBA2YUV_YV12)

#### COLOR\_XYZ2BGR public static final int COLOR\_XYZ2BGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_XYZ2BGR)

#### COLOR\_XYZ2RGB public static final int COLOR\_XYZ2RGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_XYZ2RGB)

#### COLOR\_YCrCb2BGR public static final int COLOR\_YCrCb2BGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YCrCb2BGR)

#### COLOR\_YCrCb2RGB public static final int COLOR\_YCrCb2RGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YCrCb2RGB)

#### COLOR\_YUV2BGR public static final int COLOR\_YUV2BGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGR)

#### COLOR\_YUV2BGR\_I420 public static final int COLOR\_YUV2BGR\_I420See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGR_I420)

#### COLOR\_YUV2BGR\_IYUV public static final int COLOR\_YUV2BGR\_IYUVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGR_IYUV)

#### COLOR\_YUV2BGR\_NV12 public static final int COLOR\_YUV2BGR\_NV12See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGR_NV12)

#### COLOR\_YUV2BGR\_NV21 public static final int COLOR\_YUV2BGR\_NV21See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGR_NV21)

#### COLOR\_YUV2BGR\_UYNV public static final int COLOR\_YUV2BGR\_UYNVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGR_UYNV)

#### COLOR\_YUV2BGR\_UYVY public static final int COLOR\_YUV2BGR\_UYVYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGR_UYVY)

#### COLOR\_YUV2BGR\_Y422 public static final int COLOR\_YUV2BGR\_Y422See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGR_Y422)

#### COLOR\_YUV2BGR\_YUNV public static final int COLOR\_YUV2BGR\_YUNVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGR_YUNV)

#### COLOR\_YUV2BGR\_YUY2 public static final int COLOR\_YUV2BGR\_YUY2See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGR_YUY2)

#### COLOR\_YUV2BGR\_YUYV public static final int COLOR\_YUV2BGR\_YUYVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGR_YUYV)

#### COLOR\_YUV2BGR\_YV12 public static final int COLOR\_YUV2BGR\_YV12See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGR_YV12)

#### COLOR\_YUV2BGR\_YVYU public static final int COLOR\_YUV2BGR\_YVYUSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGR_YVYU)

#### COLOR\_YUV2BGRA\_I420 public static final int COLOR\_YUV2BGRA\_I420See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGRA_I420)

#### COLOR\_YUV2BGRA\_IYUV public static final int COLOR\_YUV2BGRA\_IYUVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGRA_IYUV)

#### COLOR\_YUV2BGRA\_NV12 public static final int COLOR\_YUV2BGRA\_NV12See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGRA_NV12)

#### COLOR\_YUV2BGRA\_NV21 public static final int COLOR\_YUV2BGRA\_NV21See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGRA_NV21)

#### COLOR\_YUV2BGRA\_UYNV public static final int COLOR\_YUV2BGRA\_UYNVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGRA_UYNV)

#### COLOR\_YUV2BGRA\_UYVY public static final int COLOR\_YUV2BGRA\_UYVYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGRA_UYVY)

#### COLOR\_YUV2BGRA\_Y422 public static final int COLOR\_YUV2BGRA\_Y422See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGRA_Y422)

#### COLOR\_YUV2BGRA\_YUNV public static final int COLOR\_YUV2BGRA\_YUNVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGRA_YUNV)

#### COLOR\_YUV2BGRA\_YUY2 public static final int COLOR\_YUV2BGRA\_YUY2See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGRA_YUY2)

#### COLOR\_YUV2BGRA\_YUYV public static final int COLOR\_YUV2BGRA\_YUYVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGRA_YUYV)

#### COLOR\_YUV2BGRA\_YV12 public static final int COLOR\_YUV2BGRA\_YV12See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGRA_YV12)

#### COLOR\_YUV2BGRA\_YVYU public static final int COLOR\_YUV2BGRA\_YVYUSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2BGRA_YVYU)

#### COLOR\_YUV2GRAY\_420 public static final int COLOR\_YUV2GRAY\_420See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2GRAY_420)

#### COLOR\_YUV2GRAY\_I420 public static final int COLOR\_YUV2GRAY\_I420See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2GRAY_I420)

#### COLOR\_YUV2GRAY\_IYUV public static final int COLOR\_YUV2GRAY\_IYUVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2GRAY_IYUV)

#### COLOR\_YUV2GRAY\_NV12 public static final int COLOR\_YUV2GRAY\_NV12See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2GRAY_NV12)

#### COLOR\_YUV2GRAY\_NV21 public static final int COLOR\_YUV2GRAY\_NV21See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2GRAY_NV21)

#### COLOR\_YUV2GRAY\_UYNV public static final int COLOR\_YUV2GRAY\_UYNVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2GRAY_UYNV)

#### COLOR\_YUV2GRAY\_UYVY public static final int COLOR\_YUV2GRAY\_UYVYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2GRAY_UYVY)

#### COLOR\_YUV2GRAY\_Y422 public static final int COLOR\_YUV2GRAY\_Y422See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2GRAY_Y422)

#### COLOR\_YUV2GRAY\_YUNV public static final int COLOR\_YUV2GRAY\_YUNVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2GRAY_YUNV)

#### COLOR\_YUV2GRAY\_YUY2 public static final int COLOR\_YUV2GRAY\_YUY2See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2GRAY_YUY2)

#### COLOR\_YUV2GRAY\_YUYV public static final int COLOR\_YUV2GRAY\_YUYVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2GRAY_YUYV)

#### COLOR\_YUV2GRAY\_YV12 public static final int COLOR\_YUV2GRAY\_YV12See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2GRAY_YV12)

#### COLOR\_YUV2GRAY\_YVYU public static final int COLOR\_YUV2GRAY\_YVYUSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2GRAY_YVYU)

#### COLOR\_YUV2RGB public static final int COLOR\_YUV2RGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGB)

#### COLOR\_YUV2RGB\_I420 public static final int COLOR\_YUV2RGB\_I420See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGB_I420)

#### COLOR\_YUV2RGB\_IYUV public static final int COLOR\_YUV2RGB\_IYUVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGB_IYUV)

#### COLOR\_YUV2RGB\_NV12 public static final int COLOR\_YUV2RGB\_NV12See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGB_NV12)

#### COLOR\_YUV2RGB\_NV21 public static final int COLOR\_YUV2RGB\_NV21See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGB_NV21)

#### COLOR\_YUV2RGB\_UYNV public static final int COLOR\_YUV2RGB\_UYNVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGB_UYNV)

#### COLOR\_YUV2RGB\_UYVY public static final int COLOR\_YUV2RGB\_UYVYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGB_UYVY)

#### COLOR\_YUV2RGB\_Y422 public static final int COLOR\_YUV2RGB\_Y422See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGB_Y422)

#### COLOR\_YUV2RGB\_YUNV public static final int COLOR\_YUV2RGB\_YUNVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGB_YUNV)

#### COLOR\_YUV2RGB\_YUY2 public static final int COLOR\_YUV2RGB\_YUY2See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGB_YUY2)

#### COLOR\_YUV2RGB\_YUYV public static final int COLOR\_YUV2RGB\_YUYVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGB_YUYV)

#### COLOR\_YUV2RGB\_YV12 public static final int COLOR\_YUV2RGB\_YV12See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGB_YV12)

#### COLOR\_YUV2RGB\_YVYU public static final int COLOR\_YUV2RGB\_YVYUSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGB_YVYU)

#### COLOR\_YUV2RGBA\_I420 public static final int COLOR\_YUV2RGBA\_I420See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGBA_I420)

#### COLOR\_YUV2RGBA\_IYUV public static final int COLOR\_YUV2RGBA\_IYUVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGBA_IYUV)

#### COLOR\_YUV2RGBA\_NV12 public static final int COLOR\_YUV2RGBA\_NV12See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGBA_NV12)

#### COLOR\_YUV2RGBA\_NV21 public static final int COLOR\_YUV2RGBA\_NV21See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGBA_NV21)

#### COLOR\_YUV2RGBA\_UYNV public static final int COLOR\_YUV2RGBA\_UYNVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGBA_UYNV)

#### COLOR\_YUV2RGBA\_UYVY public static final int COLOR\_YUV2RGBA\_UYVYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGBA_UYVY)

#### COLOR\_YUV2RGBA\_Y422 public static final int COLOR\_YUV2RGBA\_Y422See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGBA_Y422)

#### COLOR\_YUV2RGBA\_YUNV public static final int COLOR\_YUV2RGBA\_YUNVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGBA_YUNV)

#### COLOR\_YUV2RGBA\_YUY2 public static final int COLOR\_YUV2RGBA\_YUY2See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGBA_YUY2)

#### COLOR\_YUV2RGBA\_YUYV public static final int COLOR\_YUV2RGBA\_YUYVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGBA_YUYV)

#### COLOR\_YUV2RGBA\_YV12 public static final int COLOR\_YUV2RGBA\_YV12See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGBA_YV12)

#### COLOR\_YUV2RGBA\_YVYU public static final int COLOR\_YUV2RGBA\_YVYUSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV2RGBA_YVYU)

#### COLOR\_YUV420p2BGR public static final int COLOR\_YUV420p2BGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV420p2BGR)

#### COLOR\_YUV420p2BGRA public static final int COLOR\_YUV420p2BGRASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV420p2BGRA)

#### COLOR\_YUV420p2GRAY public static final int COLOR\_YUV420p2GRAYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV420p2GRAY)

#### COLOR\_YUV420p2RGB public static final int COLOR\_YUV420p2RGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV420p2RGB)

#### COLOR\_YUV420p2RGBA public static final int COLOR\_YUV420p2RGBASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV420p2RGBA)

#### COLOR\_YUV420sp2BGR public static final int COLOR\_YUV420sp2BGRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV420sp2BGR)

#### COLOR\_YUV420sp2BGRA public static final int COLOR\_YUV420sp2BGRASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV420sp2BGRA)

#### COLOR\_YUV420sp2GRAY public static final int COLOR\_YUV420sp2GRAYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV420sp2GRAY)

#### COLOR\_YUV420sp2RGB public static final int COLOR\_YUV420sp2RGBSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV420sp2RGB)

#### COLOR\_YUV420sp2RGBA public static final int COLOR\_YUV420sp2RGBASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLOR_YUV420sp2RGBA)

#### COLORMAP\_AUTUMN public static final int COLORMAP\_AUTUMNSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_AUTUMN)

#### COLORMAP\_BONE public static final int COLORMAP\_BONESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_BONE)

#### COLORMAP\_CIVIDIS public static final int COLORMAP\_CIVIDISSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_CIVIDIS)

#### COLORMAP\_COOL public static final int COLORMAP\_COOLSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_COOL)

#### COLORMAP\_DEEPGREEN public static final int COLORMAP\_DEEPGREENSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_DEEPGREEN)

#### COLORMAP\_HOT public static final int COLORMAP\_HOTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_HOT)

#### COLORMAP\_HSV public static final int COLORMAP\_HSVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_HSV)

#### COLORMAP\_INFERNO public static final int COLORMAP\_INFERNOSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_INFERNO)

#### COLORMAP\_JET public static final int COLORMAP\_JETSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_JET)

#### COLORMAP\_MAGMA public static final int COLORMAP\_MAGMASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_MAGMA)

#### COLORMAP\_OCEAN public static final int COLORMAP\_OCEANSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_OCEAN)

#### COLORMAP\_PARULA public static final int COLORMAP\_PARULASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_PARULA)

#### COLORMAP\_PINK public static final int COLORMAP\_PINKSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_PINK)

#### COLORMAP\_PLASMA public static final int COLORMAP\_PLASMASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_PLASMA)

#### COLORMAP\_RAINBOW public static final int COLORMAP\_RAINBOWSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_RAINBOW)

#### COLORMAP\_SPRING public static final int COLORMAP\_SPRINGSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_SPRING)

#### COLORMAP\_SUMMER public static final int COLORMAP\_SUMMERSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_SUMMER)

#### COLORMAP\_TURBO public static final int COLORMAP\_TURBOSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_TURBO)

#### COLORMAP\_TWILIGHT public static final int COLORMAP\_TWILIGHTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_TWILIGHT)

#### COLORMAP\_TWILIGHT\_SHIFTED public static final int COLORMAP\_TWILIGHT\_SHIFTEDSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_TWILIGHT_SHIFTED)

#### COLORMAP\_VIRIDIS public static final int COLORMAP\_VIRIDISSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_VIRIDIS)

#### COLORMAP\_WINTER public static final int COLORMAP\_WINTERSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.COLORMAP_WINTER)

#### CONTOURS\_MATCH\_I1 public static final int CONTOURS\_MATCH\_I1See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CONTOURS_MATCH_I1)

#### CONTOURS\_MATCH\_I2 public static final int CONTOURS\_MATCH\_I2See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CONTOURS_MATCH_I2)

#### CONTOURS\_MATCH\_I3 public static final int CONTOURS\_MATCH\_I3See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CONTOURS_MATCH_I3)

#### CV\_BILATERAL public static final int CV\_BILATERALSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_BILATERAL)

#### CV\_BLUR public static final int CV\_BLURSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_BLUR)

#### CV\_BLUR\_NO\_SCALE public static final int CV\_BLUR\_NO\_SCALESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_BLUR_NO_SCALE)

#### CV\_CANNY\_L2\_GRADIENT public static final int CV\_CANNY\_L2\_GRADIENTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_CANNY_L2_GRADIENT)

#### CV\_CHAIN\_CODE public static final int CV\_CHAIN\_CODESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_CHAIN_CODE)

#### CV\_CLOCKWISE public static final int CV\_CLOCKWISESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_CLOCKWISE)

#### CV\_COMP\_BHATTACHARYYA public static final int CV\_COMP\_BHATTACHARYYASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_COMP_BHATTACHARYYA)

#### CV\_COMP\_CHISQR public static final int CV\_COMP\_CHISQRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_COMP_CHISQR)

#### CV\_COMP\_CHISQR\_ALT public static final int CV\_COMP\_CHISQR\_ALTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_COMP_CHISQR_ALT)

#### CV\_COMP\_CORREL public static final int CV\_COMP\_CORRELSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_COMP_CORREL)

#### CV\_COMP\_HELLINGER public static final int CV\_COMP\_HELLINGERSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_COMP_HELLINGER)

#### CV\_COMP\_INTERSECT public static final int CV\_COMP\_INTERSECTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_COMP_INTERSECT)

#### CV\_COMP\_KL\_DIV public static final int CV\_COMP\_KL\_DIVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_COMP_KL_DIV)

#### CV\_CONTOURS\_MATCH\_I1 public static final int CV\_CONTOURS\_MATCH\_I1See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_CONTOURS_MATCH_I1)

#### CV\_CONTOURS\_MATCH\_I2 public static final int CV\_CONTOURS\_MATCH\_I2See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_CONTOURS_MATCH_I2)

#### CV\_CONTOURS\_MATCH\_I3 public static final int CV\_CONTOURS\_MATCH\_I3See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_CONTOURS_MATCH_I3)

#### CV\_COUNTER\_CLOCKWISE public static final int CV\_COUNTER\_CLOCKWISESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_COUNTER_CLOCKWISE)

#### CV\_DIST\_C public static final int CV\_DIST\_CSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_DIST_C)

#### CV\_DIST\_FAIR public static final int CV\_DIST\_FAIRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_DIST_FAIR)

#### CV\_DIST\_HUBER public static final int CV\_DIST\_HUBERSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_DIST_HUBER)

#### CV\_DIST\_L1 public static final int CV\_DIST\_L1See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_DIST_L1)

#### CV\_DIST\_L12 public static final int CV\_DIST\_L12See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_DIST_L12)

#### CV\_DIST\_L2 public static final int CV\_DIST\_L2See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_DIST_L2)

#### CV\_DIST\_LABEL\_CCOMP public static final int CV\_DIST\_LABEL\_CCOMPSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_DIST_LABEL_CCOMP)

#### CV\_DIST\_LABEL\_PIXEL public static final int CV\_DIST\_LABEL\_PIXELSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_DIST_LABEL_PIXEL)

#### CV\_DIST\_MASK\_3 public static final int CV\_DIST\_MASK\_3See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_DIST_MASK_3)

#### CV\_DIST\_MASK\_5 public static final int CV\_DIST\_MASK\_5See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_DIST_MASK_5)

#### CV\_DIST\_MASK\_PRECISE public static final int CV\_DIST\_MASK\_PRECISESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_DIST_MASK_PRECISE)

#### CV\_DIST\_USER public static final int CV\_DIST\_USERSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_DIST_USER)

#### CV\_DIST\_WELSCH public static final int CV\_DIST\_WELSCHSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_DIST_WELSCH)

#### CV\_GAUSSIAN public static final int CV\_GAUSSIANSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_GAUSSIAN)

#### CV\_GAUSSIAN\_5x5 public static final int CV\_GAUSSIAN\_5x5See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_GAUSSIAN_5x5)

#### CV\_HOUGH\_GRADIENT public static final int CV\_HOUGH\_GRADIENTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_HOUGH_GRADIENT)

#### CV\_HOUGH\_MULTI\_SCALE public static final int CV\_HOUGH\_MULTI\_SCALESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_HOUGH_MULTI_SCALE)

#### CV\_HOUGH\_PROBABILISTIC public static final int CV\_HOUGH\_PROBABILISTICSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_HOUGH_PROBABILISTIC)

#### CV\_HOUGH\_STANDARD public static final int CV\_HOUGH\_STANDARDSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_HOUGH_STANDARD)

#### CV\_LINK\_RUNS public static final int CV\_LINK\_RUNSSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_LINK_RUNS)

#### CV\_MAX\_SOBEL\_KSIZE public static final int CV\_MAX\_SOBEL\_KSIZESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_MAX_SOBEL_KSIZE)

#### CV\_MEDIAN public static final int CV\_MEDIANSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_MEDIAN)

#### CV\_mRGBA2RGBA public static final int CV\_mRGBA2RGBASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_mRGBA2RGBA)

#### CV\_POLY\_APPROX\_DP public static final int CV\_POLY\_APPROX\_DPSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_POLY_APPROX_DP)

#### CV\_RGBA2mRGBA public static final int CV\_RGBA2mRGBASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_RGBA2mRGBA)

#### CV\_SCHARR public static final int CV\_SCHARRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_SCHARR)

#### CV\_SHAPE\_CROSS public static final int CV\_SHAPE\_CROSSSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_SHAPE_CROSS)

#### CV\_SHAPE\_CUSTOM public static final int CV\_SHAPE\_CUSTOMSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_SHAPE_CUSTOM)

#### CV\_SHAPE\_ELLIPSE public static final int CV\_SHAPE\_ELLIPSESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_SHAPE_ELLIPSE)

#### CV\_SHAPE\_RECT public static final int CV\_SHAPE\_RECTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_SHAPE_RECT)

#### CV\_WARP\_FILL\_OUTLIERS public static final int CV\_WARP\_FILL\_OUTLIERSSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_WARP_FILL_OUTLIERS)

#### CV\_WARP\_INVERSE\_MAP public static final int CV\_WARP\_INVERSE\_MAPSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.CV_WARP_INVERSE_MAP)

#### DIST\_C public static final int DIST\_CSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.DIST_C)

#### DIST\_FAIR public static final int DIST\_FAIRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.DIST_FAIR)

#### DIST\_HUBER public static final int DIST\_HUBERSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.DIST_HUBER)

#### DIST\_L1 public static final int DIST\_L1See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.DIST_L1)

#### DIST\_L12 public static final int DIST\_L12See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.DIST_L12)

#### DIST\_L2 public static final int DIST\_L2See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.DIST_L2)

#### DIST\_LABEL\_CCOMP public static final int DIST\_LABEL\_CCOMPSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.DIST_LABEL_CCOMP)

#### DIST\_LABEL\_PIXEL public static final int DIST\_LABEL\_PIXELSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.DIST_LABEL_PIXEL)

#### DIST\_MASK\_3 public static final int DIST\_MASK\_3See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.DIST_MASK_3)

#### DIST\_MASK\_5 public static final int DIST\_MASK\_5See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.DIST_MASK_5)

#### DIST\_MASK\_PRECISE public static final int DIST\_MASK\_PRECISESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.DIST_MASK_PRECISE)

#### DIST\_USER public static final int DIST\_USERSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.DIST_USER)

#### DIST\_WELSCH public static final int DIST\_WELSCHSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.DIST_WELSCH)

#### FLOODFILL\_FIXED\_RANGE public static final int FLOODFILL\_FIXED\_RANGESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.FLOODFILL_FIXED_RANGE)

#### FLOODFILL\_MASK\_ONLY public static final int FLOODFILL\_MASK\_ONLYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.FLOODFILL_MASK_ONLY)

#### GC\_BGD public static final int GC\_BGDSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.GC_BGD)

#### GC\_EVAL public static final int GC\_EVALSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.GC_EVAL)

#### GC\_EVAL\_FREEZE\_MODEL public static final int GC\_EVAL\_FREEZE\_MODELSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.GC_EVAL_FREEZE_MODEL)

#### GC\_FGD public static final int GC\_FGDSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.GC_FGD)

#### GC\_INIT\_WITH\_MASK public static final int GC\_INIT\_WITH\_MASKSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.GC_INIT_WITH_MASK)

#### GC\_INIT\_WITH\_RECT public static final int GC\_INIT\_WITH\_RECTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.GC_INIT_WITH_RECT)

#### GC\_PR\_BGD public static final int GC\_PR\_BGDSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.GC_PR_BGD)

#### GC\_PR\_FGD public static final int GC\_PR\_FGDSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.GC_PR_FGD)

#### HISTCMP\_BHATTACHARYYA public static final int HISTCMP\_BHATTACHARYYASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.HISTCMP_BHATTACHARYYA)

#### HISTCMP\_CHISQR public static final int HISTCMP\_CHISQRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.HISTCMP_CHISQR)

#### HISTCMP\_CHISQR\_ALT public static final int HISTCMP\_CHISQR\_ALTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.HISTCMP_CHISQR_ALT)

#### HISTCMP\_CORREL public static final int HISTCMP\_CORRELSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.HISTCMP_CORREL)

#### HISTCMP\_HELLINGER public static final int HISTCMP\_HELLINGERSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.HISTCMP_HELLINGER)

#### HISTCMP\_INTERSECT public static final int HISTCMP\_INTERSECTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.HISTCMP_INTERSECT)

#### HISTCMP\_KL\_DIV public static final int HISTCMP\_KL\_DIVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.HISTCMP_KL_DIV)

#### HOUGH\_GRADIENT public static final int HOUGH\_GRADIENTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.HOUGH_GRADIENT)

#### HOUGH\_MULTI\_SCALE public static final int HOUGH\_MULTI\_SCALESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.HOUGH_MULTI_SCALE)

#### HOUGH\_PROBABILISTIC public static final int HOUGH\_PROBABILISTICSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.HOUGH_PROBABILISTIC)

#### HOUGH\_STANDARD public static final int HOUGH\_STANDARDSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.HOUGH_STANDARD)

#### INTER\_AREA public static final int INTER\_AREASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.INTER_AREA)

#### INTER\_BITS public static final int INTER\_BITSSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.INTER_BITS)

#### INTER\_BITS2 public static final int INTER\_BITS2See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.INTER_BITS2)

#### INTER\_CUBIC public static final int INTER\_CUBICSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.INTER_CUBIC)

#### INTER\_LANCZOS4 public static final int INTER\_LANCZOS4See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.INTER_LANCZOS4)

#### INTER\_LINEAR public static final int INTER\_LINEARSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.INTER_LINEAR)

#### INTER\_LINEAR\_EXACT public static final int INTER\_LINEAR\_EXACTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.INTER_LINEAR_EXACT)

#### INTER\_MAX public static final int INTER\_MAXSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.INTER_MAX)

#### INTER\_NEAREST public static final int INTER\_NEARESTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.INTER_NEAREST)

#### INTER\_NEAREST\_EXACT public static final int INTER\_NEAREST\_EXACTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.INTER_NEAREST_EXACT)

#### INTER\_TAB\_SIZE public static final int INTER\_TAB\_SIZESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.INTER_TAB_SIZE)

#### INTER\_TAB\_SIZE2 public static final int INTER\_TAB\_SIZE2See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.INTER_TAB_SIZE2)

#### INTERSECT\_FULL public static final int INTERSECT\_FULLSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.INTERSECT_FULL)

#### INTERSECT\_NONE public static final int INTERSECT\_NONESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.INTERSECT_NONE)

#### INTERSECT\_PARTIAL public static final int INTERSECT\_PARTIALSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.INTERSECT_PARTIAL)

#### LINE\_4 public static final int LINE\_4See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.LINE_4)

#### LINE\_8 public static final int LINE\_8See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.LINE_8)

#### LINE\_AA public static final int LINE\_AASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.LINE_AA)

#### LSD\_REFINE\_ADV public static final int LSD\_REFINE\_ADVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.LSD_REFINE_ADV)

#### LSD\_REFINE\_NONE public static final int LSD\_REFINE\_NONESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.LSD_REFINE_NONE)

#### LSD\_REFINE\_STD public static final int LSD\_REFINE\_STDSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.LSD_REFINE_STD)

#### MARKER\_CROSS public static final int MARKER\_CROSSSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.MARKER_CROSS)

#### MARKER\_DIAMOND public static final int MARKER\_DIAMONDSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.MARKER_DIAMOND)

#### MARKER\_SQUARE public static final int MARKER\_SQUARESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.MARKER_SQUARE)

#### MARKER\_STAR public static final int MARKER\_STARSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.MARKER_STAR)

#### MARKER\_TILTED\_CROSS public static final int MARKER\_TILTED\_CROSSSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.MARKER_TILTED_CROSS)

#### MARKER\_TRIANGLE\_DOWN public static final int MARKER\_TRIANGLE\_DOWNSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.MARKER_TRIANGLE_DOWN)

#### MARKER\_TRIANGLE\_UP public static final int MARKER\_TRIANGLE\_UPSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.MARKER_TRIANGLE_UP)

#### MORPH\_BLACKHAT public static final int MORPH\_BLACKHATSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.MORPH_BLACKHAT)

#### MORPH\_CLOSE public static final int MORPH\_CLOSESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.MORPH_CLOSE)

#### MORPH\_CROSS public static final int MORPH\_CROSSSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.MORPH_CROSS)

#### MORPH\_DILATE public static final int MORPH\_DILATESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.MORPH_DILATE)

#### MORPH\_ELLIPSE public static final int MORPH\_ELLIPSESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.MORPH_ELLIPSE)

#### MORPH\_ERODE public static final int MORPH\_ERODESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.MORPH_ERODE)

#### MORPH\_GRADIENT public static final int MORPH\_GRADIENTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.MORPH_GRADIENT)

#### MORPH\_HITMISS public static final int MORPH\_HITMISSSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.MORPH_HITMISS)

#### MORPH\_OPEN public static final int MORPH\_OPENSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.MORPH_OPEN)

#### MORPH\_RECT public static final int MORPH\_RECTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.MORPH_RECT)

#### MORPH\_TOPHAT public static final int MORPH\_TOPHATSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.MORPH_TOPHAT)

#### PROJ\_SPHERICAL\_EQRECT public static final int PROJ\_SPHERICAL\_EQRECTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.PROJ_SPHERICAL_EQRECT)

#### PROJ\_SPHERICAL\_ORTHO public static final int PROJ\_SPHERICAL\_ORTHOSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.PROJ_SPHERICAL_ORTHO)

#### RETR\_CCOMP public static final int RETR\_CCOMPSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.RETR_CCOMP)

#### RETR\_EXTERNAL public static final int RETR\_EXTERNALSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.RETR_EXTERNAL)

#### RETR\_FLOODFILL public static final int RETR\_FLOODFILLSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.RETR_FLOODFILL)

#### RETR\_LIST public static final int RETR\_LISTSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.RETR_LIST)

#### RETR\_TREE public static final int RETR\_TREESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.RETR_TREE)

#### THRESH\_BINARY public static final int THRESH\_BINARYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.THRESH_BINARY)

#### THRESH\_BINARY\_INV public static final int THRESH\_BINARY\_INVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.THRESH_BINARY_INV)

#### THRESH\_MASK public static final int THRESH\_MASKSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.THRESH_MASK)

#### THRESH\_OTSU public static final int THRESH\_OTSUSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.THRESH_OTSU)

#### THRESH\_TOZERO public static final int THRESH\_TOZEROSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.THRESH_TOZERO)

#### THRESH\_TOZERO\_INV public static final int THRESH\_TOZERO\_INVSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.THRESH_TOZERO_INV)

#### THRESH\_TRIANGLE public static final int THRESH\_TRIANGLESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.THRESH_TRIANGLE)

#### THRESH\_TRUNC public static final int THRESH\_TRUNCSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.THRESH_TRUNC)

#### TM\_CCOEFF public static final int TM\_CCOEFFSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.TM_CCOEFF)

#### TM\_CCOEFF\_NORMED public static final int TM\_CCOEFF\_NORMEDSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.TM_CCOEFF_NORMED)

#### TM\_CCORR public static final int TM\_CCORRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.TM_CCORR)

#### TM\_CCORR\_NORMED public static final int TM\_CCORR\_NORMEDSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.TM_CCORR_NORMED)

#### TM\_SQDIFF public static final int TM\_SQDIFFSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.TM_SQDIFF)

#### TM\_SQDIFF\_NORMED public static final int TM\_SQDIFF\_NORMEDSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.TM_SQDIFF_NORMED)

#### WARP\_FILL\_OUTLIERS public static final int WARP\_FILL\_OUTLIERSSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.WARP_FILL_OUTLIERS)

#### WARP\_INVERSE\_MAP public static final int WARP\_INVERSE\_MAPSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.WARP_INVERSE_MAP)

#### WARP\_POLAR\_LINEAR public static final int WARP\_POLAR\_LINEARSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.WARP_POLAR_LINEAR)

#### WARP\_POLAR\_LOG public static final int WARP\_POLAR\_LOGSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.imgproc.Imgproc.WARP_POLAR_LOG)

### Constructor Detail

#### Imgproc public Imgproc()

### Method Detail

#### accumulate public static void accumulate([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Adds an image to the accumulator image. The function adds src or some of its elements to dst : \(\texttt{dst} (x,y) \leftarrow \texttt{dst} (x,y) + \texttt{src} (x,y) \quad \text{if} \quad \texttt{mask} (x,y) \ne 0\) The function supports multi-channel images. Each channel is processed independently. The function cv::accumulate can be used, for example, to collect statistics of a scene background viewed by a still camera and for the further foreground-background segmentation.Parameters:src - Input image of type CV\_8UC(n), CV\_16UC(n), CV\_32FC(n) or CV\_64FC(n), where n is a positive integer.dst - %Accumulator image with the same number of channels as input image, and a depth of CV\_32F or CV\_64F. SEE: accumulateSquare, accumulateProduct, accumulateWeighted

#### accumulate public static void accumulate([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask) Adds an image to the accumulator image. The function adds src or some of its elements to dst : \(\texttt{dst} (x,y) \leftarrow \texttt{dst} (x,y) + \texttt{src} (x,y) \quad \text{if} \quad \texttt{mask} (x,y) \ne 0\) The function supports multi-channel images. Each channel is processed independently. The function cv::accumulate can be used, for example, to collect statistics of a scene background viewed by a still camera and for the further foreground-background segmentation.Parameters:src - Input image of type CV\_8UC(n), CV\_16UC(n), CV\_32FC(n) or CV\_64FC(n), where n is a positive integer.dst - %Accumulator image with the same number of channels as input image, and a depth of CV\_32F or CV\_64F.mask - Optional operation mask. SEE: accumulateSquare, accumulateProduct, accumulateWeighted

#### accumulateProduct public static void accumulateProduct([Mat](http://docs.google.com/org/opencv/core/Mat.html) src1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) src2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Adds the per-element product of two input images to the accumulator image. The function adds the product of two images or their selected regions to the accumulator dst : \(\texttt{dst} (x,y) \leftarrow \texttt{dst} (x,y) + \texttt{src1} (x,y) \cdot \texttt{src2} (x,y) \quad \text{if} \quad \texttt{mask} (x,y) \ne 0\) The function supports multi-channel images. Each channel is processed independently.Parameters:src1 - First input image, 1- or 3-channel, 8-bit or 32-bit floating point.src2 - Second input image of the same type and the same size as src1 .dst - %Accumulator image with the same number of channels as input images, 32-bit or 64-bit floating-point. SEE: accumulate, accumulateSquare, accumulateWeighted

#### accumulateProduct public static void accumulateProduct([Mat](http://docs.google.com/org/opencv/core/Mat.html) src1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) src2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask) Adds the per-element product of two input images to the accumulator image. The function adds the product of two images or their selected regions to the accumulator dst : \(\texttt{dst} (x,y) \leftarrow \texttt{dst} (x,y) + \texttt{src1} (x,y) \cdot \texttt{src2} (x,y) \quad \text{if} \quad \texttt{mask} (x,y) \ne 0\) The function supports multi-channel images. Each channel is processed independently.Parameters:src1 - First input image, 1- or 3-channel, 8-bit or 32-bit floating point.src2 - Second input image of the same type and the same size as src1 .dst - %Accumulator image with the same number of channels as input images, 32-bit or 64-bit floating-point.mask - Optional operation mask. SEE: accumulate, accumulateSquare, accumulateWeighted

#### accumulateSquare public static void accumulateSquare([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Adds the square of a source image to the accumulator image. The function adds the input image src or its selected region, raised to a power of 2, to the accumulator dst : \(\texttt{dst} (x,y) \leftarrow \texttt{dst} (x,y) + \texttt{src} (x,y)^2 \quad \text{if} \quad \texttt{mask} (x,y) \ne 0\) The function supports multi-channel images. Each channel is processed independently.Parameters:src - Input image as 1- or 3-channel, 8-bit or 32-bit floating point.dst - %Accumulator image with the same number of channels as input image, 32-bit or 64-bit floating-point. SEE: accumulateSquare, accumulateProduct, accumulateWeighted

#### accumulateSquare public static void accumulateSquare([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask) Adds the square of a source image to the accumulator image. The function adds the input image src or its selected region, raised to a power of 2, to the accumulator dst : \(\texttt{dst} (x,y) \leftarrow \texttt{dst} (x,y) + \texttt{src} (x,y)^2 \quad \text{if} \quad \texttt{mask} (x,y) \ne 0\) The function supports multi-channel images. Each channel is processed independently.Parameters:src - Input image as 1- or 3-channel, 8-bit or 32-bit floating point.dst - %Accumulator image with the same number of channels as input image, 32-bit or 64-bit floating-point.mask - Optional operation mask. SEE: accumulateSquare, accumulateProduct, accumulateWeighted

#### accumulateWeighted public static void accumulateWeighted([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, double alpha) Updates a running average. The function calculates the weighted sum of the input image src and the accumulator dst so that dst becomes a running average of a frame sequence: \(\texttt{dst} (x,y) \leftarrow (1- \texttt{alpha} ) \cdot \texttt{dst} (x,y) + \texttt{alpha} \cdot \texttt{src} (x,y) \quad \text{if} \quad \texttt{mask} (x,y) \ne 0\) That is, alpha regulates the update speed (how fast the accumulator "forgets" about earlier images). The function supports multi-channel images. Each channel is processed independently.Parameters:src - Input image as 1- or 3-channel, 8-bit or 32-bit floating point.dst - %Accumulator image with the same number of channels as input image, 32-bit or 64-bit floating-point.alpha - Weight of the input image. SEE: accumulate, accumulateSquare, accumulateProduct

#### accumulateWeighted public static void accumulateWeighted([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, double alpha, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask) Updates a running average. The function calculates the weighted sum of the input image src and the accumulator dst so that dst becomes a running average of a frame sequence: \(\texttt{dst} (x,y) \leftarrow (1- \texttt{alpha} ) \cdot \texttt{dst} (x,y) + \texttt{alpha} \cdot \texttt{src} (x,y) \quad \text{if} \quad \texttt{mask} (x,y) \ne 0\) That is, alpha regulates the update speed (how fast the accumulator "forgets" about earlier images). The function supports multi-channel images. Each channel is processed independently.Parameters:src - Input image as 1- or 3-channel, 8-bit or 32-bit floating point.dst - %Accumulator image with the same number of channels as input image, 32-bit or 64-bit floating-point.alpha - Weight of the input image.mask - Optional operation mask. SEE: accumulate, accumulateSquare, accumulateProduct

#### adaptiveThreshold public static void adaptiveThreshold([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, double maxValue, int adaptiveMethod, int thresholdType, int blockSize, double C)

Applies an adaptive threshold to an array. The function transforms a grayscale image to a binary image according to the formulae:

* + - * **THRESH\_BINARY** \(dst(x,y) = \fork{\texttt{maxValue}}{if \(src(x,y) > T(x,y)\)}{0}{otherwise}\)
      * **THRESH\_BINARY\_INV** \(dst(x,y) = \fork{0}{if \(src(x,y) > T(x,y)\)}{\texttt{maxValue}}{otherwise}\) where \(T(x,y)\) is a threshold calculated individually for each pixel (see adaptiveMethod parameter).

The function can process the image in-place.Parameters:src - Source 8-bit single-channel image.dst - Destination image of the same size and the same type as src.maxValue - Non-zero value assigned to the pixels for which the condition is satisfiedadaptiveMethod - Adaptive thresholding algorithm to use, see #AdaptiveThresholdTypes. The #BORDER\_REPLICATE | #BORDER\_ISOLATED is used to process boundaries.thresholdType - Thresholding type that must be either #THRESH\_BINARY or #THRESH\_BINARY\_INV, see #ThresholdTypes.blockSize - Size of a pixel neighborhood that is used to calculate a threshold value for the pixel: 3, 5, 7, and so on.C - Constant subtracted from the mean or weighted mean (see the details below). Normally, it is positive but may be zero or negative as well. SEE: threshold, blur, GaussianBlur

#### applyColorMap public static void applyColorMap([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int colormap) Applies a GNU Octave/MATLAB equivalent colormap on a given image.Parameters:src - The source image, grayscale or colored of type CV\_8UC1 or CV\_8UC3.dst - The result is the colormapped source image. Note: Mat::create is called on dst.colormap - The colormap to apply, see #ColormapTypes

#### applyColorMap public static void applyColorMap([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) userColor) Applies a user colormap on a given image.Parameters:src - The source image, grayscale or colored of type CV\_8UC1 or CV\_8UC3.dst - The result is the colormapped source image. Note: Mat::create is called on dst.userColor - The colormap to apply of type CV\_8UC1 or CV\_8UC3 and size 256

#### approxPolyDP public static void approxPolyDP([MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) curve, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) approxCurve, double epsilon, boolean closed) Approximates a polygonal curve(s) with the specified precision. The function cv::approxPolyDP approximates a curve or a polygon with another curve/polygon with less vertices so that the distance between them is less or equal to the specified precision. It uses the Douglas-Peucker algorithm <http://en.wikipedia.org/wiki/Ramer-Douglas-Peucker\_algorithm>Parameters:curve - Input vector of a 2D point stored in std::vector or MatapproxCurve - Result of the approximation. The type should match the type of the input curve.epsilon - Parameter specifying the approximation accuracy. This is the maximum distance between the original curve and its approximation.closed - If true, the approximated curve is closed (its first and last vertices are connected). Otherwise, it is not closed.

#### arcLength public static double arcLength([MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) curve, boolean closed) Calculates a contour perimeter or a curve length. The function computes a curve length or a closed contour perimeter.Parameters:curve - Input vector of 2D points, stored in std::vector or Mat.closed - Flag indicating whether the curve is closed or not. Returns:automatically generated

#### arrowedLine public static void arrowedLine([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Draws a arrow segment pointing from the first point to the second one. The function cv::arrowedLine draws an arrow between pt1 and pt2 points in the image. See also #line.Parameters:img - Image.pt1 - The point the arrow starts from.pt2 - The point the arrow points to.color - Line color.

#### arrowedLine public static void arrowedLine([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness) Draws a arrow segment pointing from the first point to the second one. The function cv::arrowedLine draws an arrow between pt1 and pt2 points in the image. See also #line.Parameters:img - Image.pt1 - The point the arrow starts from.pt2 - The point the arrow points to.color - Line color.thickness - Line thickness.

#### arrowedLine public static void arrowedLine([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int line\_type) Draws a arrow segment pointing from the first point to the second one. The function cv::arrowedLine draws an arrow between pt1 and pt2 points in the image. See also #line.Parameters:img - Image.pt1 - The point the arrow starts from.pt2 - The point the arrow points to.color - Line color.thickness - Line thickness.line\_type - Type of the line. See #LineTypes

#### arrowedLine public static void arrowedLine([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int line\_type, int shift) Draws a arrow segment pointing from the first point to the second one. The function cv::arrowedLine draws an arrow between pt1 and pt2 points in the image. See also #line.Parameters:img - Image.pt1 - The point the arrow starts from.pt2 - The point the arrow points to.color - Line color.thickness - Line thickness.line\_type - Type of the line. See #LineTypesshift - Number of fractional bits in the point coordinates.

#### arrowedLine public static void arrowedLine([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int line\_type, int shift, double tipLength) Draws a arrow segment pointing from the first point to the second one. The function cv::arrowedLine draws an arrow between pt1 and pt2 points in the image. See also #line.Parameters:img - Image.pt1 - The point the arrow starts from.pt2 - The point the arrow points to.color - Line color.thickness - Line thickness.line\_type - Type of the line. See #LineTypesshift - Number of fractional bits in the point coordinates.tipLength - The length of the arrow tip in relation to the arrow length

#### bilateralFilter public static void bilateralFilter([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int d, double sigmaColor, double sigmaSpace) Applies the bilateral filter to an image. The function applies bilateral filtering to the input image, as described in http://www.dai.ed.ac.uk/CVonline/LOCAL\_COPIES/MANDUCHI1/Bilateral\_Filtering.html bilateralFilter can reduce unwanted noise very well while keeping edges fairly sharp. However, it is very slow compared to most filters. \_Sigma values\_: For simplicity, you can set the 2 sigma values to be the same. If they are small (< 10), the filter will not have much effect, whereas if they are large (> 150), they will have a very strong effect, making the image look "cartoonish". \_Filter size\_: Large filters (d > 5) are very slow, so it is recommended to use d=5 for real-time applications, and perhaps d=9 for offline applications that need heavy noise filtering. This filter does not work inplace.Parameters:src - Source 8-bit or floating-point, 1-channel or 3-channel image.dst - Destination image of the same size and type as src .d - Diameter of each pixel neighborhood that is used during filtering. If it is non-positive, it is computed from sigmaSpace.sigmaColor - Filter sigma in the color space. A larger value of the parameter means that farther colors within the pixel neighborhood (see sigmaSpace) will be mixed together, resulting in larger areas of semi-equal color.sigmaSpace - Filter sigma in the coordinate space. A larger value of the parameter means that farther pixels will influence each other as long as their colors are close enough (see sigmaColor ). When d>0, it specifies the neighborhood size regardless of sigmaSpace. Otherwise, d is proportional to sigmaSpace.

#### bilateralFilter public static void bilateralFilter([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int d, double sigmaColor, double sigmaSpace, int borderType) Applies the bilateral filter to an image. The function applies bilateral filtering to the input image, as described in http://www.dai.ed.ac.uk/CVonline/LOCAL\_COPIES/MANDUCHI1/Bilateral\_Filtering.html bilateralFilter can reduce unwanted noise very well while keeping edges fairly sharp. However, it is very slow compared to most filters. \_Sigma values\_: For simplicity, you can set the 2 sigma values to be the same. If they are small (< 10), the filter will not have much effect, whereas if they are large (> 150), they will have a very strong effect, making the image look "cartoonish". \_Filter size\_: Large filters (d > 5) are very slow, so it is recommended to use d=5 for real-time applications, and perhaps d=9 for offline applications that need heavy noise filtering. This filter does not work inplace.Parameters:src - Source 8-bit or floating-point, 1-channel or 3-channel image.dst - Destination image of the same size and type as src .d - Diameter of each pixel neighborhood that is used during filtering. If it is non-positive, it is computed from sigmaSpace.sigmaColor - Filter sigma in the color space. A larger value of the parameter means that farther colors within the pixel neighborhood (see sigmaSpace) will be mixed together, resulting in larger areas of semi-equal color.sigmaSpace - Filter sigma in the coordinate space. A larger value of the parameter means that farther pixels will influence each other as long as their colors are close enough (see sigmaColor ). When d>0, it specifies the neighborhood size regardless of sigmaSpace. Otherwise, d is proportional to sigmaSpace.borderType - border mode used to extrapolate pixels outside of the image, see #BorderTypes

#### blendLinear public static void blendLinear([Mat](http://docs.google.com/org/opencv/core/Mat.html) src1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) src2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) weights1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) weights2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst)

#### blur public static void blur([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize) Blurs an image using the normalized box filter. The function smooths an image using the kernel: \(\texttt{K} = \frac{1}{\texttt{ksize.width\*ksize.height}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ \hdotsfor{6} \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ \end{bmatrix}\) The call blur(src, dst, ksize, anchor, borderType) is equivalent to `boxFilter(src, dst, src.type(), ksize, anchor, true, borderType)`.Parameters:src - input image; it can have any number of channels, which are processed independently, but the depth should be CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - output image of the same size and type as src.ksize - blurring kernel size. center. SEE: boxFilter, bilateralFilter, GaussianBlur, medianBlur

#### blur public static void blur([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor) Blurs an image using the normalized box filter. The function smooths an image using the kernel: \(\texttt{K} = \frac{1}{\texttt{ksize.width\*ksize.height}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ \hdotsfor{6} \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ \end{bmatrix}\) The call blur(src, dst, ksize, anchor, borderType) is equivalent to `boxFilter(src, dst, src.type(), ksize, anchor, true, borderType)`.Parameters:src - input image; it can have any number of channels, which are processed independently, but the depth should be CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - output image of the same size and type as src.ksize - blurring kernel size.anchor - anchor point; default value Point(-1,-1) means that the anchor is at the kernel center. SEE: boxFilter, bilateralFilter, GaussianBlur, medianBlur

#### blur public static void blur([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int borderType) Blurs an image using the normalized box filter. The function smooths an image using the kernel: \(\texttt{K} = \frac{1}{\texttt{ksize.width\*ksize.height}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ \hdotsfor{6} \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ \end{bmatrix}\) The call blur(src, dst, ksize, anchor, borderType) is equivalent to `boxFilter(src, dst, src.type(), ksize, anchor, true, borderType)`.Parameters:src - input image; it can have any number of channels, which are processed independently, but the depth should be CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - output image of the same size and type as src.ksize - blurring kernel size.anchor - anchor point; default value Point(-1,-1) means that the anchor is at the kernel center.borderType - border mode used to extrapolate pixels outside of the image, see #BorderTypes. #BORDER\_WRAP is not supported. SEE: boxFilter, bilateralFilter, GaussianBlur, medianBlur

#### boundingRect public static [Rect](http://docs.google.com/org/opencv/core/Rect.html) boundingRect([Mat](http://docs.google.com/org/opencv/core/Mat.html) array) Calculates the up-right bounding rectangle of a point set or non-zero pixels of gray-scale image. The function calculates and returns the minimal up-right bounding rectangle for the specified point set or non-zero pixels of gray-scale image.Parameters:array - Input gray-scale image or 2D point set, stored in std::vector or Mat. Returns:automatically generated

#### boxFilter public static void boxFilter([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize) Blurs an image using the box filter. The function smooths an image using the kernel: \(\texttt{K} = \alpha \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ \hdotsfor{6} \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}\) where \(\alpha = \begin{cases} \frac{1}{\texttt{ksize.width\*ksize.height}} & \texttt{when } \texttt{normalize=true} \\1 & \texttt{otherwise}\end{cases}\) Unnormalized box filter is useful for computing various integral characteristics over each pixel neighborhood, such as covariance matrices of image derivatives (used in dense optical flow algorithms, and so on). If you need to compute pixel sums over variable-size windows, use #integral.Parameters:src - input image.dst - output image of the same size and type as src.ddepth - the output image depth (-1 to use src.depth()).ksize - blurring kernel size. center. SEE: blur, bilateralFilter, GaussianBlur, medianBlur, integral

#### boxFilter public static void boxFilter([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor) Blurs an image using the box filter. The function smooths an image using the kernel: \(\texttt{K} = \alpha \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ \hdotsfor{6} \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}\) where \(\alpha = \begin{cases} \frac{1}{\texttt{ksize.width\*ksize.height}} & \texttt{when } \texttt{normalize=true} \\1 & \texttt{otherwise}\end{cases}\) Unnormalized box filter is useful for computing various integral characteristics over each pixel neighborhood, such as covariance matrices of image derivatives (used in dense optical flow algorithms, and so on). If you need to compute pixel sums over variable-size windows, use #integral.Parameters:src - input image.dst - output image of the same size and type as src.ddepth - the output image depth (-1 to use src.depth()).ksize - blurring kernel size.anchor - anchor point; default value Point(-1,-1) means that the anchor is at the kernel center. SEE: blur, bilateralFilter, GaussianBlur, medianBlur, integral

#### boxFilter public static void boxFilter([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, boolean normalize) Blurs an image using the box filter. The function smooths an image using the kernel: \(\texttt{K} = \alpha \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ \hdotsfor{6} \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}\) where \(\alpha = \begin{cases} \frac{1}{\texttt{ksize.width\*ksize.height}} & \texttt{when } \texttt{normalize=true} \\1 & \texttt{otherwise}\end{cases}\) Unnormalized box filter is useful for computing various integral characteristics over each pixel neighborhood, such as covariance matrices of image derivatives (used in dense optical flow algorithms, and so on). If you need to compute pixel sums over variable-size windows, use #integral.Parameters:src - input image.dst - output image of the same size and type as src.ddepth - the output image depth (-1 to use src.depth()).ksize - blurring kernel size.anchor - anchor point; default value Point(-1,-1) means that the anchor is at the kernel center.normalize - flag, specifying whether the kernel is normalized by its area or not. SEE: blur, bilateralFilter, GaussianBlur, medianBlur, integral

#### boxFilter public static void boxFilter([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, boolean normalize, int borderType) Blurs an image using the box filter. The function smooths an image using the kernel: \(\texttt{K} = \alpha \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ \hdotsfor{6} \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}\) where \(\alpha = \begin{cases} \frac{1}{\texttt{ksize.width\*ksize.height}} & \texttt{when } \texttt{normalize=true} \\1 & \texttt{otherwise}\end{cases}\) Unnormalized box filter is useful for computing various integral characteristics over each pixel neighborhood, such as covariance matrices of image derivatives (used in dense optical flow algorithms, and so on). If you need to compute pixel sums over variable-size windows, use #integral.Parameters:src - input image.dst - output image of the same size and type as src.ddepth - the output image depth (-1 to use src.depth()).ksize - blurring kernel size.anchor - anchor point; default value Point(-1,-1) means that the anchor is at the kernel center.normalize - flag, specifying whether the kernel is normalized by its area or not.borderType - border mode used to extrapolate pixels outside of the image, see #BorderTypes. #BORDER\_WRAP is not supported. SEE: blur, bilateralFilter, GaussianBlur, medianBlur, integral

#### boxPoints public static void boxPoints([RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) box, [Mat](http://docs.google.com/org/opencv/core/Mat.html) points) Finds the four vertices of a rotated rect. Useful to draw the rotated rectangle. The function finds the four vertices of a rotated rectangle. This function is useful to draw the rectangle. In C++, instead of using this function, you can directly use RotatedRect::points method. Please visit the REF: tutorial\_bounding\_rotated\_ellipses "tutorial on Creating Bounding rotated boxes and ellipses for contours" for more information.Parameters:box - The input rotated rectangle. It may be the output ofpoints - The output array of four vertices of rectangles.

#### calcBackProject public static void calcBackProject(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> images, [MatOfInt](http://docs.google.com/org/opencv/core/MatOfInt.html) channels, [Mat](http://docs.google.com/org/opencv/core/Mat.html) hist, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) ranges, double scale)

#### calcHist public static void calcHist(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> images, [MatOfInt](http://docs.google.com/org/opencv/core/MatOfInt.html) channels, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) hist, [MatOfInt](http://docs.google.com/org/opencv/core/MatOfInt.html) histSize, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) ranges)

#### calcHist public static void calcHist(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> images, [MatOfInt](http://docs.google.com/org/opencv/core/MatOfInt.html) channels, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) hist, [MatOfInt](http://docs.google.com/org/opencv/core/MatOfInt.html) histSize, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) ranges, boolean accumulate)

#### Canny public static void Canny([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) edges, double threshold1, double threshold2) Finds edges in an image using the Canny algorithm CITE: Canny86 . The function finds edges in the input image and marks them in the output map edges using the Canny algorithm. The smallest value between threshold1 and threshold2 is used for edge linking. The largest value is used to find initial segments of strong edges. See <http://en.wikipedia.org/wiki/Canny\_edge\_detector>Parameters:image - 8-bit input image.edges - output edge map; single channels 8-bit image, which has the same size as image .threshold1 - first threshold for the hysteresis procedure.threshold2 - second threshold for the hysteresis procedure. \(=\sqrt{(dI/dx)^2 + (dI/dy)^2}\) should be used to calculate the image gradient magnitude ( L2gradient=true ), or whether the default \(L\_1\) norm \(=|dI/dx|+|dI/dy|\) is enough ( L2gradient=false ).

#### Canny public static void Canny([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) edges, double threshold1, double threshold2, int apertureSize) Finds edges in an image using the Canny algorithm CITE: Canny86 . The function finds edges in the input image and marks them in the output map edges using the Canny algorithm. The smallest value between threshold1 and threshold2 is used for edge linking. The largest value is used to find initial segments of strong edges. See <http://en.wikipedia.org/wiki/Canny\_edge\_detector>Parameters:image - 8-bit input image.edges - output edge map; single channels 8-bit image, which has the same size as image .threshold1 - first threshold for the hysteresis procedure.threshold2 - second threshold for the hysteresis procedure.apertureSize - aperture size for the Sobel operator. \(=\sqrt{(dI/dx)^2 + (dI/dy)^2}\) should be used to calculate the image gradient magnitude ( L2gradient=true ), or whether the default \(L\_1\) norm \(=|dI/dx|+|dI/dy|\) is enough ( L2gradient=false ).

#### Canny public static void Canny([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) edges, double threshold1, double threshold2, int apertureSize, boolean L2gradient) Finds edges in an image using the Canny algorithm CITE: Canny86 . The function finds edges in the input image and marks them in the output map edges using the Canny algorithm. The smallest value between threshold1 and threshold2 is used for edge linking. The largest value is used to find initial segments of strong edges. See <http://en.wikipedia.org/wiki/Canny\_edge\_detector>Parameters:image - 8-bit input image.edges - output edge map; single channels 8-bit image, which has the same size as image .threshold1 - first threshold for the hysteresis procedure.threshold2 - second threshold for the hysteresis procedure.apertureSize - aperture size for the Sobel operator.L2gradient - a flag, indicating whether a more accurate \(L\_2\) norm \(=\sqrt{(dI/dx)^2 + (dI/dy)^2}\) should be used to calculate the image gradient magnitude ( L2gradient=true ), or whether the default \(L\_1\) norm \(=|dI/dx|+|dI/dy|\) is enough ( L2gradient=false ).

#### Canny public static void Canny([Mat](http://docs.google.com/org/opencv/core/Mat.html) dx, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dy, [Mat](http://docs.google.com/org/opencv/core/Mat.html) edges, double threshold1, double threshold2) \overload Finds edges in an image using the Canny algorithm with custom image gradient.Parameters:dx - 16-bit x derivative of input image (CV\_16SC1 or CV\_16SC3).dy - 16-bit y derivative of input image (same type as dx).edges - output edge map; single channels 8-bit image, which has the same size as image .threshold1 - first threshold for the hysteresis procedure.threshold2 - second threshold for the hysteresis procedure. \(=\sqrt{(dI/dx)^2 + (dI/dy)^2}\) should be used to calculate the image gradient magnitude ( L2gradient=true ), or whether the default \(L\_1\) norm \(=|dI/dx|+|dI/dy|\) is enough ( L2gradient=false ).

#### Canny public static void Canny([Mat](http://docs.google.com/org/opencv/core/Mat.html) dx, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dy, [Mat](http://docs.google.com/org/opencv/core/Mat.html) edges, double threshold1, double threshold2, boolean L2gradient) \overload Finds edges in an image using the Canny algorithm with custom image gradient.Parameters:dx - 16-bit x derivative of input image (CV\_16SC1 or CV\_16SC3).dy - 16-bit y derivative of input image (same type as dx).edges - output edge map; single channels 8-bit image, which has the same size as image .threshold1 - first threshold for the hysteresis procedure.threshold2 - second threshold for the hysteresis procedure.L2gradient - a flag, indicating whether a more accurate \(L\_2\) norm \(=\sqrt{(dI/dx)^2 + (dI/dy)^2}\) should be used to calculate the image gradient magnitude ( L2gradient=true ), or whether the default \(L\_1\) norm \(=|dI/dx|+|dI/dy|\) is enough ( L2gradient=false ).

#### circle public static void circle([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) center, int radius, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Draws a circle. The function cv::circle draws a simple or filled circle with a given center and radius.Parameters:img - Image where the circle is drawn.center - Center of the circle.radius - Radius of the circle.color - Circle color. mean that a filled circle is to be drawn.

#### circle public static void circle([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) center, int radius, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness) Draws a circle. The function cv::circle draws a simple or filled circle with a given center and radius.Parameters:img - Image where the circle is drawn.center - Center of the circle.radius - Radius of the circle.color - Circle color.thickness - Thickness of the circle outline, if positive. Negative values, like #FILLED, mean that a filled circle is to be drawn.

#### circle public static void circle([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) center, int radius, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType) Draws a circle. The function cv::circle draws a simple or filled circle with a given center and radius.Parameters:img - Image where the circle is drawn.center - Center of the circle.radius - Radius of the circle.color - Circle color.thickness - Thickness of the circle outline, if positive. Negative values, like #FILLED, mean that a filled circle is to be drawn.lineType - Type of the circle boundary. See #LineTypes

#### circle public static void circle([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) center, int radius, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType, int shift) Draws a circle. The function cv::circle draws a simple or filled circle with a given center and radius.Parameters:img - Image where the circle is drawn.center - Center of the circle.radius - Radius of the circle.color - Circle color.thickness - Thickness of the circle outline, if positive. Negative values, like #FILLED, mean that a filled circle is to be drawn.lineType - Type of the circle boundary. See #LineTypesshift - Number of fractional bits in the coordinates of the center and in the radius value.

#### clipLine public static boolean clipLine([Rect](http://docs.google.com/org/opencv/core/Rect.html) imgRect, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2)Parameters:imgRect - Image rectangle.pt1 - First line point.pt2 - Second line point. Returns:automatically generated

#### compareHist public static double compareHist([Mat](http://docs.google.com/org/opencv/core/Mat.html) H1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) H2, int method) Compares two histograms. The function cv::compareHist compares two dense or two sparse histograms using the specified method. The function returns \(d(H\_1, H\_2)\) . While the function works well with 1-, 2-, 3-dimensional dense histograms, it may not be suitable for high-dimensional sparse histograms. In such histograms, because of aliasing and sampling problems, the coordinates of non-zero histogram bins can slightly shift. To compare such histograms or more general sparse configurations of weighted points, consider using the #EMD function.Parameters:H1 - First compared histogram.H2 - Second compared histogram of the same size as H1 .method - Comparison method, see #HistCompMethods Returns:automatically generated

#### connectedComponents public static int connectedComponents([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels)Parameters:image - the 8-bit single-channel image to be labeledlabels - destination labeled image Returns:automatically generated

#### connectedComponents public static int connectedComponents([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels, int connectivity)Parameters:image - the 8-bit single-channel image to be labeledlabels - destination labeled imageconnectivity - 8 or 4 for 8-way or 4-way connectivity respectively Returns:automatically generated

#### connectedComponents public static int connectedComponents([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels, int connectivity, int ltype)Parameters:image - the 8-bit single-channel image to be labeledlabels - destination labeled imageconnectivity - 8 or 4 for 8-way or 4-way connectivity respectivelyltype - output image label type. Currently CV\_32S and CV\_16U are supported. Returns:automatically generated

#### connectedComponentsWithAlgorithm public static int connectedComponentsWithAlgorithm([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels, int connectivity, int ltype, int ccltype) computes the connected components labeled image of boolean image image with 4 or 8 way connectivity - returns N, the total number of labels [0, N-1] where 0 represents the background label. ltype specifies the output label image type, an important consideration based on the total number of labels or alternatively the total number of pixels in the source image. ccltype specifies the connected components labeling algorithm to use, currently Grana (BBDT) and Wu's (SAUF) CITE: Wu2009 algorithms are supported, see the #ConnectedComponentsAlgorithmsTypes for details. Note that SAUF algorithm forces a row major ordering of labels while BBDT does not. This function uses parallel version of both Grana and Wu's algorithms if at least one allowed parallel framework is enabled and if the rows of the image are at least twice the number returned by #getNumberOfCPUs.Parameters:image - the 8-bit single-channel image to be labeledlabels - destination labeled imageconnectivity - 8 or 4 for 8-way or 4-way connectivity respectivelyltype - output image label type. Currently CV\_32S and CV\_16U are supported.ccltype - connected components algorithm type (see the #ConnectedComponentsAlgorithmsTypes). Returns:automatically generated

#### connectedComponentsWithStats public static int connectedComponentsWithStats([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels, [Mat](http://docs.google.com/org/opencv/core/Mat.html) stats, [Mat](http://docs.google.com/org/opencv/core/Mat.html) centroids)Parameters:image - the 8-bit single-channel image to be labeledlabels - destination labeled imagestats - statistics output for each label, including the background label. Statistics are accessed via stats(label, COLUMN) where COLUMN is one of #ConnectedComponentsTypes, selecting the statistic. The data type is CV\_32S.centroids - centroid output for each label, including the background label. Centroids are accessed via centroids(label, 0) for x and centroids(label, 1) for y. The data type CV\_64F. Returns:automatically generated

#### connectedComponentsWithStats public static int connectedComponentsWithStats([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels, [Mat](http://docs.google.com/org/opencv/core/Mat.html) stats, [Mat](http://docs.google.com/org/opencv/core/Mat.html) centroids, int connectivity)Parameters:image - the 8-bit single-channel image to be labeledlabels - destination labeled imagestats - statistics output for each label, including the background label. Statistics are accessed via stats(label, COLUMN) where COLUMN is one of #ConnectedComponentsTypes, selecting the statistic. The data type is CV\_32S.centroids - centroid output for each label, including the background label. Centroids are accessed via centroids(label, 0) for x and centroids(label, 1) for y. The data type CV\_64F.connectivity - 8 or 4 for 8-way or 4-way connectivity respectively Returns:automatically generated

#### connectedComponentsWithStats public static int connectedComponentsWithStats([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels, [Mat](http://docs.google.com/org/opencv/core/Mat.html) stats, [Mat](http://docs.google.com/org/opencv/core/Mat.html) centroids, int connectivity, int ltype)Parameters:image - the 8-bit single-channel image to be labeledlabels - destination labeled imagestats - statistics output for each label, including the background label. Statistics are accessed via stats(label, COLUMN) where COLUMN is one of #ConnectedComponentsTypes, selecting the statistic. The data type is CV\_32S.centroids - centroid output for each label, including the background label. Centroids are accessed via centroids(label, 0) for x and centroids(label, 1) for y. The data type CV\_64F.connectivity - 8 or 4 for 8-way or 4-way connectivity respectivelyltype - output image label type. Currently CV\_32S and CV\_16U are supported. Returns:automatically generated

#### connectedComponentsWithStatsWithAlgorithm public static int connectedComponentsWithStatsWithAlgorithm([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels, [Mat](http://docs.google.com/org/opencv/core/Mat.html) stats, [Mat](http://docs.google.com/org/opencv/core/Mat.html) centroids, int connectivity, int ltype, int ccltype) computes the connected components labeled image of boolean image and also produces a statistics output for each label image with 4 or 8 way connectivity - returns N, the total number of labels [0, N-1] where 0 represents the background label. ltype specifies the output label image type, an important consideration based on the total number of labels or alternatively the total number of pixels in the source image. ccltype specifies the connected components labeling algorithm to use, currently Grana's (BBDT) and Wu's (SAUF) CITE: Wu2009 algorithms are supported, see the #ConnectedComponentsAlgorithmsTypes for details. Note that SAUF algorithm forces a row major ordering of labels while BBDT does not. This function uses parallel version of both Grana and Wu's algorithms (statistics included) if at least one allowed parallel framework is enabled and if the rows of the image are at least twice the number returned by #getNumberOfCPUs.Parameters:image - the 8-bit single-channel image to be labeledlabels - destination labeled imagestats - statistics output for each label, including the background label. Statistics are accessed via stats(label, COLUMN) where COLUMN is one of #ConnectedComponentsTypes, selecting the statistic. The data type is CV\_32S.centroids - centroid output for each label, including the background label. Centroids are accessed via centroids(label, 0) for x and centroids(label, 1) for y. The data type CV\_64F.connectivity - 8 or 4 for 8-way or 4-way connectivity respectivelyltype - output image label type. Currently CV\_32S and CV\_16U are supported.ccltype - connected components algorithm type (see #ConnectedComponentsAlgorithmsTypes). Returns:automatically generated

#### contourArea public static double contourArea([Mat](http://docs.google.com/org/opencv/core/Mat.html) contour) Calculates a contour area. The function computes a contour area. Similarly to moments , the area is computed using the Green formula. Thus, the returned area and the number of non-zero pixels, if you draw the contour using #drawContours or #fillPoly , can be different. Also, the function will most certainly give a wrong results for contours with self-intersections. Example: vector<Point> contour; contour.push\_back(Point2f(0, 0)); contour.push\_back(Point2f(10, 0)); contour.push\_back(Point2f(10, 10)); contour.push\_back(Point2f(5, 4)); double area0 = contourArea(contour); vector<Point> approx; approxPolyDP(contour, approx, 5, true); double area1 = contourArea(approx); cout << "area0 =" << area0 << endl << "area1 =" << area1 << endl << "approx poly vertices" << approx.size() << endl;Parameters:contour - Input vector of 2D points (contour vertices), stored in std::vector or Mat. depending on the contour orientation (clockwise or counter-clockwise). Using this feature you can determine orientation of a contour by taking the sign of an area. By default, the parameter is false, which means that the absolute value is returned. Returns:automatically generated

#### contourArea public static double contourArea([Mat](http://docs.google.com/org/opencv/core/Mat.html) contour, boolean oriented) Calculates a contour area. The function computes a contour area. Similarly to moments , the area is computed using the Green formula. Thus, the returned area and the number of non-zero pixels, if you draw the contour using #drawContours or #fillPoly , can be different. Also, the function will most certainly give a wrong results for contours with self-intersections. Example: vector<Point> contour; contour.push\_back(Point2f(0, 0)); contour.push\_back(Point2f(10, 0)); contour.push\_back(Point2f(10, 10)); contour.push\_back(Point2f(5, 4)); double area0 = contourArea(contour); vector<Point> approx; approxPolyDP(contour, approx, 5, true); double area1 = contourArea(approx); cout << "area0 =" << area0 << endl << "area1 =" << area1 << endl << "approx poly vertices" << approx.size() << endl;Parameters:contour - Input vector of 2D points (contour vertices), stored in std::vector or Mat.oriented - Oriented area flag. If it is true, the function returns a signed area value, depending on the contour orientation (clockwise or counter-clockwise). Using this feature you can determine orientation of a contour by taking the sign of an area. By default, the parameter is false, which means that the absolute value is returned. Returns:automatically generated

#### convertMaps public static void convertMaps([Mat](http://docs.google.com/org/opencv/core/Mat.html) map1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dstmap1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dstmap2, int dstmap1type)

Converts image transformation maps from one representation to another. The function converts a pair of maps for remap from one representation to another. The following options ( (map1.type(), map2.type()) \(\rightarrow\) (dstmap1.type(), dstmap2.type()) ) are supported:

* + - * \(\texttt{(CV\_32FC1, CV\_32FC1)} \rightarrow \texttt{(CV\_16SC2, CV\_16UC1)}\). This is the most frequently used conversion operation, in which the original floating-point maps (see remap ) are converted to a more compact and much faster fixed-point representation. The first output array contains the rounded coordinates and the second array (created only when nninterpolation=false ) contains indices in the interpolation tables.
      * \(\texttt{(CV\_32FC2)} \rightarrow \texttt{(CV\_16SC2, CV\_16UC1)}\). The same as above but the original maps are stored in one 2-channel matrix.
      * Reverse conversion. Obviously, the reconstructed floating-point maps will not be exactly the same as the originals.

Parameters:map1 - The first input map of type CV\_16SC2, CV\_32FC1, or CV\_32FC2 .map2 - The second input map of type CV\_16UC1, CV\_32FC1, or none (empty matrix), respectively.dstmap1 - The first output map that has the type dstmap1type and the same size as src .dstmap2 - The second output map.dstmap1type - Type of the first output map that should be CV\_16SC2, CV\_32FC1, or CV\_32FC2 . nearest-neighbor or for a more complex interpolation. SEE: remap, undistort, initUndistortRectifyMap

#### convertMaps public static void convertMaps([Mat](http://docs.google.com/org/opencv/core/Mat.html) map1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dstmap1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dstmap2, int dstmap1type, boolean nninterpolation)

Converts image transformation maps from one representation to another. The function converts a pair of maps for remap from one representation to another. The following options ( (map1.type(), map2.type()) \(\rightarrow\) (dstmap1.type(), dstmap2.type()) ) are supported:

* + - * \(\texttt{(CV\_32FC1, CV\_32FC1)} \rightarrow \texttt{(CV\_16SC2, CV\_16UC1)}\). This is the most frequently used conversion operation, in which the original floating-point maps (see remap ) are converted to a more compact and much faster fixed-point representation. The first output array contains the rounded coordinates and the second array (created only when nninterpolation=false ) contains indices in the interpolation tables.
      * \(\texttt{(CV\_32FC2)} \rightarrow \texttt{(CV\_16SC2, CV\_16UC1)}\). The same as above but the original maps are stored in one 2-channel matrix.
      * Reverse conversion. Obviously, the reconstructed floating-point maps will not be exactly the same as the originals.

Parameters:map1 - The first input map of type CV\_16SC2, CV\_32FC1, or CV\_32FC2 .map2 - The second input map of type CV\_16UC1, CV\_32FC1, or none (empty matrix), respectively.dstmap1 - The first output map that has the type dstmap1type and the same size as src .dstmap2 - The second output map.dstmap1type - Type of the first output map that should be CV\_16SC2, CV\_32FC1, or CV\_32FC2 .nninterpolation - Flag indicating whether the fixed-point maps are used for the nearest-neighbor or for a more complex interpolation. SEE: remap, undistort, initUndistortRectifyMap

#### convexHull public static void convexHull([MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) points, [MatOfInt](http://docs.google.com/org/opencv/core/MatOfInt.html) hull) Finds the convex hull of a point set. The function cv::convexHull finds the convex hull of a 2D point set using the Sklansky's algorithm CITE: Sklansky82 that has \*O(N logN)\* complexity in the current implementation.Parameters:points - Input 2D point set, stored in std::vector or Mat.hull - Output convex hull. It is either an integer vector of indices or vector of points. In the first case, the hull elements are 0-based indices of the convex hull points in the original array (since the set of convex hull points is a subset of the original point set). In the second case, hull elements are the convex hull points themselves. Otherwise, it is oriented counter-clockwise. The assumed coordinate system has its X axis pointing to the right, and its Y axis pointing upwards. returns convex hull points. Otherwise, it returns indices of the convex hull points. When the output array is std::vector, the flag is ignored, and the output depends on the type of the vector: std::vector<int> implies returnPoints=false, std::vector<Point> implies returnPoints=true. **Note:** points and hull should be different arrays, inplace processing isn't supported. Check REF: tutorial\_hull "the corresponding tutorial" for more details. useful links: https://www.learnopencv.com/convex-hull-using-opencv-in-python-and-c/

#### convexHull public static void convexHull([MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) points, [MatOfInt](http://docs.google.com/org/opencv/core/MatOfInt.html) hull, boolean clockwise) Finds the convex hull of a point set. The function cv::convexHull finds the convex hull of a 2D point set using the Sklansky's algorithm CITE: Sklansky82 that has \*O(N logN)\* complexity in the current implementation.Parameters:points - Input 2D point set, stored in std::vector or Mat.hull - Output convex hull. It is either an integer vector of indices or vector of points. In the first case, the hull elements are 0-based indices of the convex hull points in the original array (since the set of convex hull points is a subset of the original point set). In the second case, hull elements are the convex hull points themselves.clockwise - Orientation flag. If it is true, the output convex hull is oriented clockwise. Otherwise, it is oriented counter-clockwise. The assumed coordinate system has its X axis pointing to the right, and its Y axis pointing upwards. returns convex hull points. Otherwise, it returns indices of the convex hull points. When the output array is std::vector, the flag is ignored, and the output depends on the type of the vector: std::vector<int> implies returnPoints=false, std::vector<Point> implies returnPoints=true. **Note:** points and hull should be different arrays, inplace processing isn't supported. Check REF: tutorial\_hull "the corresponding tutorial" for more details. useful links: https://www.learnopencv.com/convex-hull-using-opencv-in-python-and-c/

#### convexityDefects public static void convexityDefects([MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) contour, [MatOfInt](http://docs.google.com/org/opencv/core/MatOfInt.html) convexhull, [MatOfInt4](http://docs.google.com/org/opencv/core/MatOfInt4.html) convexityDefects) Finds the convexity defects of a contour. The figure below displays convexity defects of a hand contour: ![image](pics/defects.png)Parameters:contour - Input contour.convexhull - Convex hull obtained using convexHull that should contain indices of the contour points that make the hull.convexityDefects - The output vector of convexity defects. In C++ and the new Python/Java interface each convexity defect is represented as 4-element integer vector (a.k.a. #Vec4i): (start\_index, end\_index, farthest\_pt\_index, fixpt\_depth), where indices are 0-based indices in the original contour of the convexity defect beginning, end and the farthest point, and fixpt\_depth is fixed-point approximation (with 8 fractional bits) of the distance between the farthest contour point and the hull. That is, to get the floating-point value of the depth will be fixpt\_depth/256.0.

#### cornerEigenValsAndVecs public static void cornerEigenValsAndVecs([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int blockSize, int ksize)

Calculates eigenvalues and eigenvectors of image blocks for corner detection. For every pixel \(p\) , the function cornerEigenValsAndVecs considers a blockSize \(\times\) blockSize neighborhood \(S(p)\) . It calculates the covariation matrix of derivatives over the neighborhood as: \(M = \begin{bmatrix} \sum \_{S(p)}(dI/dx)^2 & \sum \_{S(p)}dI/dx dI/dy \\ \sum \_{S(p)}dI/dx dI/dy & \sum \_{S(p)}(dI/dy)^2 \end{bmatrix}\) where the derivatives are computed using the Sobel operator. After that, it finds eigenvectors and eigenvalues of \(M\) and stores them in the destination image as \((\lambda\_1, \lambda\_2, x\_1, y\_1, x\_2, y\_2)\) where

* + - * \(\lambda\_1, \lambda\_2\) are the non-sorted eigenvalues of \(M\)
      * \(x\_1, y\_1\) are the eigenvectors corresponding to \(\lambda\_1\)
      * \(x\_2, y\_2\) are the eigenvectors corresponding to \(\lambda\_2\)

The output of the function can be used for robust edge or corner detection.Parameters:src - Input single-channel 8-bit or floating-point image.dst - Image to store the results. It has the same size as src and the type CV\_32FC(6) .blockSize - Neighborhood size (see details below).ksize - Aperture parameter for the Sobel operator. SEE: cornerMinEigenVal, cornerHarris, preCornerDetect

#### cornerEigenValsAndVecs public static void cornerEigenValsAndVecs([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int blockSize, int ksize, int borderType)

Calculates eigenvalues and eigenvectors of image blocks for corner detection. For every pixel \(p\) , the function cornerEigenValsAndVecs considers a blockSize \(\times\) blockSize neighborhood \(S(p)\) . It calculates the covariation matrix of derivatives over the neighborhood as: \(M = \begin{bmatrix} \sum \_{S(p)}(dI/dx)^2 & \sum \_{S(p)}dI/dx dI/dy \\ \sum \_{S(p)}dI/dx dI/dy & \sum \_{S(p)}(dI/dy)^2 \end{bmatrix}\) where the derivatives are computed using the Sobel operator. After that, it finds eigenvectors and eigenvalues of \(M\) and stores them in the destination image as \((\lambda\_1, \lambda\_2, x\_1, y\_1, x\_2, y\_2)\) where

* + - * \(\lambda\_1, \lambda\_2\) are the non-sorted eigenvalues of \(M\)
      * \(x\_1, y\_1\) are the eigenvectors corresponding to \(\lambda\_1\)
      * \(x\_2, y\_2\) are the eigenvectors corresponding to \(\lambda\_2\)

The output of the function can be used for robust edge or corner detection.Parameters:src - Input single-channel 8-bit or floating-point image.dst - Image to store the results. It has the same size as src and the type CV\_32FC(6) .blockSize - Neighborhood size (see details below).ksize - Aperture parameter for the Sobel operator.borderType - Pixel extrapolation method. See #BorderTypes. #BORDER\_WRAP is not supported. SEE: cornerMinEigenVal, cornerHarris, preCornerDetect

#### cornerHarris public static void cornerHarris([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int blockSize, int ksize, double k) Harris corner detector. The function runs the Harris corner detector on the image. Similarly to cornerMinEigenVal and cornerEigenValsAndVecs , for each pixel \((x, y)\) it calculates a \(2\times2\) gradient covariance matrix \(M^{(x,y)}\) over a \(\texttt{blockSize} \times \texttt{blockSize}\) neighborhood. Then, it computes the following characteristic: \(\texttt{dst} (x,y) = \mathrm{det} M^{(x,y)} - k \cdot \left ( \mathrm{tr} M^{(x,y)} \right )^2\) Corners in the image can be found as the local maxima of this response map.Parameters:src - Input single-channel 8-bit or floating-point image.dst - Image to store the Harris detector responses. It has the type CV\_32FC1 and the same size as src .blockSize - Neighborhood size (see the details on #cornerEigenValsAndVecs ).ksize - Aperture parameter for the Sobel operator.k - Harris detector free parameter. See the formula above.

#### cornerHarris public static void cornerHarris([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int blockSize, int ksize, double k, int borderType) Harris corner detector. The function runs the Harris corner detector on the image. Similarly to cornerMinEigenVal and cornerEigenValsAndVecs , for each pixel \((x, y)\) it calculates a \(2\times2\) gradient covariance matrix \(M^{(x,y)}\) over a \(\texttt{blockSize} \times \texttt{blockSize}\) neighborhood. Then, it computes the following characteristic: \(\texttt{dst} (x,y) = \mathrm{det} M^{(x,y)} - k \cdot \left ( \mathrm{tr} M^{(x,y)} \right )^2\) Corners in the image can be found as the local maxima of this response map.Parameters:src - Input single-channel 8-bit or floating-point image.dst - Image to store the Harris detector responses. It has the type CV\_32FC1 and the same size as src .blockSize - Neighborhood size (see the details on #cornerEigenValsAndVecs ).ksize - Aperture parameter for the Sobel operator.k - Harris detector free parameter. See the formula above.borderType - Pixel extrapolation method. See #BorderTypes. #BORDER\_WRAP is not supported.

#### cornerMinEigenVal public static void cornerMinEigenVal([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int blockSize) Calculates the minimal eigenvalue of gradient matrices for corner detection. The function is similar to cornerEigenValsAndVecs but it calculates and stores only the minimal eigenvalue of the covariance matrix of derivatives, that is, \(\min(\lambda\_1, \lambda\_2)\) in terms of the formulae in the cornerEigenValsAndVecs description.Parameters:src - Input single-channel 8-bit or floating-point image.dst - Image to store the minimal eigenvalues. It has the type CV\_32FC1 and the same size as src .blockSize - Neighborhood size (see the details on #cornerEigenValsAndVecs ).

#### cornerMinEigenVal public static void cornerMinEigenVal([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int blockSize, int ksize) Calculates the minimal eigenvalue of gradient matrices for corner detection. The function is similar to cornerEigenValsAndVecs but it calculates and stores only the minimal eigenvalue of the covariance matrix of derivatives, that is, \(\min(\lambda\_1, \lambda\_2)\) in terms of the formulae in the cornerEigenValsAndVecs description.Parameters:src - Input single-channel 8-bit or floating-point image.dst - Image to store the minimal eigenvalues. It has the type CV\_32FC1 and the same size as src .blockSize - Neighborhood size (see the details on #cornerEigenValsAndVecs ).ksize - Aperture parameter for the Sobel operator.

#### cornerMinEigenVal public static void cornerMinEigenVal([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int blockSize, int ksize, int borderType) Calculates the minimal eigenvalue of gradient matrices for corner detection. The function is similar to cornerEigenValsAndVecs but it calculates and stores only the minimal eigenvalue of the covariance matrix of derivatives, that is, \(\min(\lambda\_1, \lambda\_2)\) in terms of the formulae in the cornerEigenValsAndVecs description.Parameters:src - Input single-channel 8-bit or floating-point image.dst - Image to store the minimal eigenvalues. It has the type CV\_32FC1 and the same size as src .blockSize - Neighborhood size (see the details on #cornerEigenValsAndVecs ).ksize - Aperture parameter for the Sobel operator.borderType - Pixel extrapolation method. See #BorderTypes. #BORDER\_WRAP is not supported.

#### cornerSubPix public static void cornerSubPix([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) corners, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, [Size](http://docs.google.com/org/opencv/core/Size.html) zeroZone, [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) criteria) Refines the corner locations. The function iterates to find the sub-pixel accurate location of corners or radial saddle points as described in CITE: forstner1987fast, and as shown on the figure below. ![image](pics/cornersubpix.png) Sub-pixel accurate corner locator is based on the observation that every vector from the center \(q\) to a point \(p\) located within a neighborhood of \(q\) is orthogonal to the image gradient at \(p\) subject to image and measurement noise. Consider the expression: \(\epsilon \_i = {DI\_{p\_i}}^T \cdot (q - p\_i)\) where \({DI\_{p\_i}}\) is an image gradient at one of the points \(p\_i\) in a neighborhood of \(q\) . The value of \(q\) is to be found so that \(\epsilon\_i\) is minimized. A system of equations may be set up with \(\epsilon\_i\) set to zero: \(\sum \_i(DI\_{p\_i} \cdot {DI\_{p\_i}}^T) \cdot q - \sum \_i(DI\_{p\_i} \cdot {DI\_{p\_i}}^T \cdot p\_i)\) where the gradients are summed within a neighborhood ("search window") of \(q\) . Calling the first gradient term \(G\) and the second gradient term \(b\) gives: \(q = G^{-1} \cdot b\) The algorithm sets the center of the neighborhood window at this new center \(q\) and then iterates until the center stays within a set threshold.Parameters:image - Input single-channel, 8-bit or float image.corners - Initial coordinates of the input corners and refined coordinates provided for output.winSize - Half of the side length of the search window. For example, if winSize=Size(5,5) , then a \((5\*2+1) \times (5\*2+1) = 11 \times 11\) search window is used.zeroZone - Half of the size of the dead region in the middle of the search zone over which the summation in the formula below is not done. It is used sometimes to avoid possible singularities of the autocorrelation matrix. The value of (-1,-1) indicates that there is no such a size.criteria - Criteria for termination of the iterative process of corner refinement. That is, the process of corner position refinement stops either after criteria.maxCount iterations or when the corner position moves by less than criteria.epsilon on some iteration.

#### createCLAHE public static [CLAHE](http://docs.google.com/org/opencv/imgproc/CLAHE.html) createCLAHE() Creates a smart pointer to a cv::CLAHE class and initializes it. equally sized rectangular tiles. tileGridSize defines the number of tiles in row and column.Returns:automatically generated

#### createCLAHE public static [CLAHE](http://docs.google.com/org/opencv/imgproc/CLAHE.html) createCLAHE(double clipLimit) Creates a smart pointer to a cv::CLAHE class and initializes it.Parameters:clipLimit - Threshold for contrast limiting. equally sized rectangular tiles. tileGridSize defines the number of tiles in row and column. Returns:automatically generated

#### createCLAHE public static [CLAHE](http://docs.google.com/org/opencv/imgproc/CLAHE.html) createCLAHE(double clipLimit, [Size](http://docs.google.com/org/opencv/core/Size.html) tileGridSize) Creates a smart pointer to a cv::CLAHE class and initializes it.Parameters:clipLimit - Threshold for contrast limiting.tileGridSize - Size of grid for histogram equalization. Input image will be divided into equally sized rectangular tiles. tileGridSize defines the number of tiles in row and column. Returns:automatically generated

#### createGeneralizedHoughBallard public static [GeneralizedHoughBallard](http://docs.google.com/org/opencv/imgproc/GeneralizedHoughBallard.html) createGeneralizedHoughBallard() Creates a smart pointer to a cv::GeneralizedHoughBallard class and initializes it.Returns:automatically generated

#### createGeneralizedHoughGuil public static [GeneralizedHoughGuil](http://docs.google.com/org/opencv/imgproc/GeneralizedHoughGuil.html) createGeneralizedHoughGuil() Creates a smart pointer to a cv::GeneralizedHoughGuil class and initializes it.Returns:automatically generated

#### createHanningWindow public static void createHanningWindow([Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) winSize, int type) This function computes a Hanning window coefficients in two dimensions. See (http://en.wikipedia.org/wiki/Hann\_function) and (http://en.wikipedia.org/wiki/Window\_function) for more information. An example is shown below: // create hanning window of size 100x100 and type CV\_32F Mat hann; createHanningWindow(hann, Size(100, 100), CV\_32F);Parameters:dst - Destination array to place Hann coefficients inwinSize - The window size specifications (both width and height must be > 1)type - Created array type

#### createLineSegmentDetector public static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) createLineSegmentDetector() Creates a smart pointer to a LineSegmentDetector object and initializes it. The LineSegmentDetector algorithm is defined using the standard values. Only advanced users may want to edit those, as to tailor it for their own application. is chosen. **Note:** Implementation has been removed due original code license conflictReturns:automatically generated

#### createLineSegmentDetector public static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) createLineSegmentDetector(int \_refine) Creates a smart pointer to a LineSegmentDetector object and initializes it. The LineSegmentDetector algorithm is defined using the standard values. Only advanced users may want to edit those, as to tailor it for their own application.Parameters:\_refine - The way found lines will be refined, see #LineSegmentDetectorModes is chosen. **Note:** Implementation has been removed due original code license conflict Returns:automatically generated

#### createLineSegmentDetector public static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) createLineSegmentDetector(int \_refine, double \_scale) Creates a smart pointer to a LineSegmentDetector object and initializes it. The LineSegmentDetector algorithm is defined using the standard values. Only advanced users may want to edit those, as to tailor it for their own application.Parameters:\_refine - The way found lines will be refined, see #LineSegmentDetectorModes\_scale - The scale of the image that will be used to find the lines. Range (0..1]. is chosen. **Note:** Implementation has been removed due original code license conflict Returns:automatically generated

#### createLineSegmentDetector public static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) createLineSegmentDetector(int \_refine, double \_scale, double \_sigma\_scale) Creates a smart pointer to a LineSegmentDetector object and initializes it. The LineSegmentDetector algorithm is defined using the standard values. Only advanced users may want to edit those, as to tailor it for their own application.Parameters:\_refine - The way found lines will be refined, see #LineSegmentDetectorModes\_scale - The scale of the image that will be used to find the lines. Range (0..1].\_sigma\_scale - Sigma for Gaussian filter. It is computed as sigma = \_sigma\_scale/\_scale. is chosen. **Note:** Implementation has been removed due original code license conflict Returns:automatically generated

#### createLineSegmentDetector public static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) createLineSegmentDetector(int \_refine, double \_scale, double \_sigma\_scale, double \_quant) Creates a smart pointer to a LineSegmentDetector object and initializes it. The LineSegmentDetector algorithm is defined using the standard values. Only advanced users may want to edit those, as to tailor it for their own application.Parameters:\_refine - The way found lines will be refined, see #LineSegmentDetectorModes\_scale - The scale of the image that will be used to find the lines. Range (0..1].\_sigma\_scale - Sigma for Gaussian filter. It is computed as sigma = \_sigma\_scale/\_scale.\_quant - Bound to the quantization error on the gradient norm. is chosen. **Note:** Implementation has been removed due original code license conflict Returns:automatically generated

#### createLineSegmentDetector public static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) createLineSegmentDetector(int \_refine, double \_scale, double \_sigma\_scale, double \_quant, double \_ang\_th) Creates a smart pointer to a LineSegmentDetector object and initializes it. The LineSegmentDetector algorithm is defined using the standard values. Only advanced users may want to edit those, as to tailor it for their own application.Parameters:\_refine - The way found lines will be refined, see #LineSegmentDetectorModes\_scale - The scale of the image that will be used to find the lines. Range (0..1].\_sigma\_scale - Sigma for Gaussian filter. It is computed as sigma = \_sigma\_scale/\_scale.\_quant - Bound to the quantization error on the gradient norm.\_ang\_th - Gradient angle tolerance in degrees. is chosen. **Note:** Implementation has been removed due original code license conflict Returns:automatically generated

#### createLineSegmentDetector public static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) createLineSegmentDetector(int \_refine, double \_scale, double \_sigma\_scale, double \_quant, double \_ang\_th, double \_log\_eps) Creates a smart pointer to a LineSegmentDetector object and initializes it. The LineSegmentDetector algorithm is defined using the standard values. Only advanced users may want to edit those, as to tailor it for their own application.Parameters:\_refine - The way found lines will be refined, see #LineSegmentDetectorModes\_scale - The scale of the image that will be used to find the lines. Range (0..1].\_sigma\_scale - Sigma for Gaussian filter. It is computed as sigma = \_sigma\_scale/\_scale.\_quant - Bound to the quantization error on the gradient norm.\_ang\_th - Gradient angle tolerance in degrees.\_log\_eps - Detection threshold: -log10(NFA) > log\_eps. Used only when advance refinement is chosen. **Note:** Implementation has been removed due original code license conflict Returns:automatically generated

#### createLineSegmentDetector public static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) createLineSegmentDetector(int \_refine, double \_scale, double \_sigma\_scale, double \_quant, double \_ang\_th, double \_log\_eps, double \_density\_th) Creates a smart pointer to a LineSegmentDetector object and initializes it. The LineSegmentDetector algorithm is defined using the standard values. Only advanced users may want to edit those, as to tailor it for their own application.Parameters:\_refine - The way found lines will be refined, see #LineSegmentDetectorModes\_scale - The scale of the image that will be used to find the lines. Range (0..1].\_sigma\_scale - Sigma for Gaussian filter. It is computed as sigma = \_sigma\_scale/\_scale.\_quant - Bound to the quantization error on the gradient norm.\_ang\_th - Gradient angle tolerance in degrees.\_log\_eps - Detection threshold: -log10(NFA) > log\_eps. Used only when advance refinement is chosen.\_density\_th - Minimal density of aligned region points in the enclosing rectangle. **Note:** Implementation has been removed due original code license conflict Returns:automatically generated

#### createLineSegmentDetector public static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) createLineSegmentDetector(int \_refine, double \_scale, double \_sigma\_scale, double \_quant, double \_ang\_th, double \_log\_eps, double \_density\_th, int \_n\_bins) Creates a smart pointer to a LineSegmentDetector object and initializes it. The LineSegmentDetector algorithm is defined using the standard values. Only advanced users may want to edit those, as to tailor it for their own application.Parameters:\_refine - The way found lines will be refined, see #LineSegmentDetectorModes\_scale - The scale of the image that will be used to find the lines. Range (0..1].\_sigma\_scale - Sigma for Gaussian filter. It is computed as sigma = \_sigma\_scale/\_scale.\_quant - Bound to the quantization error on the gradient norm.\_ang\_th - Gradient angle tolerance in degrees.\_log\_eps - Detection threshold: -log10(NFA) > log\_eps. Used only when advance refinement is chosen.\_density\_th - Minimal density of aligned region points in the enclosing rectangle.\_n\_bins - Number of bins in pseudo-ordering of gradient modulus. **Note:** Implementation has been removed due original code license conflict Returns:automatically generated

#### cvtColor public static void cvtColor([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int code)

Converts an image from one color space to another. The function converts an input image from one color space to another. In case of a transformation to-from RGB color space, the order of the channels should be specified explicitly (RGB or BGR). Note that the default color format in OpenCV is often referred to as RGB but it is actually BGR (the bytes are reversed). So the first byte in a standard (24-bit) color image will be an 8-bit Blue component, the second byte will be Green, and the third byte will be Red. The fourth, fifth, and sixth bytes would then be the second pixel (Blue, then Green, then Red), and so on. The conventional ranges for R, G, and B channel values are:

* + - * 0 to 255 for CV\_8U images
      * 0 to 65535 for CV\_16U images
      * 0 to 1 for CV\_32F images

In case of linear transformations, the range does not matter. But in case of a non-linear transformation, an input RGB image should be normalized to the proper value range to get the correct results, for example, for RGB \(\rightarrow\) L\\*u\\*v\\* transformation. For example, if you have a 32-bit floating-point image directly converted from an 8-bit image without any scaling, then it will have the 0..255 value range instead of 0..1 assumed by the function. So, before calling #cvtColor , you need first to scale the image down: img \*= 1./255; cvtColor(img, img, COLOR\_BGR2Luv); If you use #cvtColor with 8-bit images, the conversion will have some information lost. For many applications, this will not be noticeable but it is recommended to use 32-bit images in applications that need the full range of colors or that convert an image before an operation and then convert back. If conversion adds the alpha channel, its value will set to the maximum of corresponding channel range: 255 for CV\_8U, 65535 for CV\_16U, 1 for CV\_32F.Parameters:src - input image: 8-bit unsigned, 16-bit unsigned ( CV\_16UC... ), or single-precision floating-point.dst - output image of the same size and depth as src.code - color space conversion code (see #ColorConversionCodes). channels is derived automatically from src and code. SEE: REF: imgproc\_color\_conversions

#### cvtColor public static void cvtColor([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int code, int dstCn)

Converts an image from one color space to another. The function converts an input image from one color space to another. In case of a transformation to-from RGB color space, the order of the channels should be specified explicitly (RGB or BGR). Note that the default color format in OpenCV is often referred to as RGB but it is actually BGR (the bytes are reversed). So the first byte in a standard (24-bit) color image will be an 8-bit Blue component, the second byte will be Green, and the third byte will be Red. The fourth, fifth, and sixth bytes would then be the second pixel (Blue, then Green, then Red), and so on. The conventional ranges for R, G, and B channel values are:

* + - * 0 to 255 for CV\_8U images
      * 0 to 65535 for CV\_16U images
      * 0 to 1 for CV\_32F images

In case of linear transformations, the range does not matter. But in case of a non-linear transformation, an input RGB image should be normalized to the proper value range to get the correct results, for example, for RGB \(\rightarrow\) L\\*u\\*v\\* transformation. For example, if you have a 32-bit floating-point image directly converted from an 8-bit image without any scaling, then it will have the 0..255 value range instead of 0..1 assumed by the function. So, before calling #cvtColor , you need first to scale the image down: img \*= 1./255; cvtColor(img, img, COLOR\_BGR2Luv); If you use #cvtColor with 8-bit images, the conversion will have some information lost. For many applications, this will not be noticeable but it is recommended to use 32-bit images in applications that need the full range of colors or that convert an image before an operation and then convert back. If conversion adds the alpha channel, its value will set to the maximum of corresponding channel range: 255 for CV\_8U, 65535 for CV\_16U, 1 for CV\_32F.Parameters:src - input image: 8-bit unsigned, 16-bit unsigned ( CV\_16UC... ), or single-precision floating-point.dst - output image of the same size and depth as src.code - color space conversion code (see #ColorConversionCodes).dstCn - number of channels in the destination image; if the parameter is 0, the number of the channels is derived automatically from src and code. SEE: REF: imgproc\_color\_conversions

#### cvtColorTwoPlane public static void cvtColorTwoPlane([Mat](http://docs.google.com/org/opencv/core/Mat.html) src1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) src2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int code)

Converts an image from one color space to another where the source image is stored in two planes. This function only supports YUV420 to RGB conversion as of now.

* + - * #COLOR\_YUV2BGR\_NV12
      * #COLOR\_YUV2RGB\_NV12
      * #COLOR\_YUV2BGRA\_NV12
      * #COLOR\_YUV2RGBA\_NV12
      * #COLOR\_YUV2BGR\_NV21
      * #COLOR\_YUV2RGB\_NV21
      * #COLOR\_YUV2BGRA\_NV21
      * #COLOR\_YUV2RGBA\_NV21

Parameters:src1 - automatically generatedsrc2 - automatically generateddst - automatically generatedcode - automatically generated

#### demosaicing public static void demosaicing([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int code) main function for all demosaicing processesParameters:src - input image: 8-bit unsigned or 16-bit unsigned.dst - output image of the same size and depth as src.code - Color space conversion code (see the description below). channels is derived automatically from src and code. The function can do the following transformations:

* + - * Demosaicing using bilinear interpolation

#COLOR\_BayerBG2BGR , #COLOR\_BayerGB2BGR , #COLOR\_BayerRG2BGR , #COLOR\_BayerGR2BGR #COLOR\_BayerBG2GRAY , #COLOR\_BayerGB2GRAY , #COLOR\_BayerRG2GRAY , #COLOR\_BayerGR2GRAY

* + - * Demosaicing using Variable Number of Gradients.

#COLOR\_BayerBG2BGR\_VNG , #COLOR\_BayerGB2BGR\_VNG , #COLOR\_BayerRG2BGR\_VNG , #COLOR\_BayerGR2BGR\_VNG

* + - * Edge-Aware Demosaicing.

#COLOR\_BayerBG2BGR\_EA , #COLOR\_BayerGB2BGR\_EA , #COLOR\_BayerRG2BGR\_EA , #COLOR\_BayerGR2BGR\_EA

* + - * Demosaicing with alpha channel

#COLOR\_BayerBG2BGRA , #COLOR\_BayerGB2BGRA , #COLOR\_BayerRG2BGRA , #COLOR\_BayerGR2BGRA SEE: cvtColor

#### demosaicing public static void demosaicing([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int code, int dstCn) main function for all demosaicing processesParameters:src - input image: 8-bit unsigned or 16-bit unsigned.dst - output image of the same size and depth as src.code - Color space conversion code (see the description below).dstCn - number of channels in the destination image; if the parameter is 0, the number of the channels is derived automatically from src and code. The function can do the following transformations:

* + - * Demosaicing using bilinear interpolation

#COLOR\_BayerBG2BGR , #COLOR\_BayerGB2BGR , #COLOR\_BayerRG2BGR , #COLOR\_BayerGR2BGR #COLOR\_BayerBG2GRAY , #COLOR\_BayerGB2GRAY , #COLOR\_BayerRG2GRAY , #COLOR\_BayerGR2GRAY

* + - * Demosaicing using Variable Number of Gradients.

#COLOR\_BayerBG2BGR\_VNG , #COLOR\_BayerGB2BGR\_VNG , #COLOR\_BayerRG2BGR\_VNG , #COLOR\_BayerGR2BGR\_VNG

* + - * Edge-Aware Demosaicing.

#COLOR\_BayerBG2BGR\_EA , #COLOR\_BayerGB2BGR\_EA , #COLOR\_BayerRG2BGR\_EA , #COLOR\_BayerGR2BGR\_EA

* + - * Demosaicing with alpha channel

#COLOR\_BayerBG2BGRA , #COLOR\_BayerGB2BGRA , #COLOR\_BayerRG2BGRA , #COLOR\_BayerGR2BGRA SEE: cvtColor

#### dilate public static void dilate([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel) Dilates an image by using a specific structuring element. The function dilates the source image using the specified structuring element that determines the shape of a pixel neighborhood over which the maximum is taken: \(\texttt{dst} (x,y) = \max \_{(x',y'): \, \texttt{element} (x',y') \ne0 } \texttt{src} (x+x',y+y')\) The function supports the in-place mode. Dilation can be applied several ( iterations ) times. In case of multi-channel images, each channel is processed independently.Parameters:src - input image; the number of channels can be arbitrary, but the depth should be one of CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - output image of the same size and type as src.kernel - structuring element used for dilation; if elemenat=Mat(), a 3 x 3 rectangular structuring element is used. Kernel can be created using #getStructuringElement anchor is at the element center. SEE: erode, morphologyEx, getStructuringElement

#### dilate public static void dilate([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor) Dilates an image by using a specific structuring element. The function dilates the source image using the specified structuring element that determines the shape of a pixel neighborhood over which the maximum is taken: \(\texttt{dst} (x,y) = \max \_{(x',y'): \, \texttt{element} (x',y') \ne0 } \texttt{src} (x+x',y+y')\) The function supports the in-place mode. Dilation can be applied several ( iterations ) times. In case of multi-channel images, each channel is processed independently.Parameters:src - input image; the number of channels can be arbitrary, but the depth should be one of CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - output image of the same size and type as src.kernel - structuring element used for dilation; if elemenat=Mat(), a 3 x 3 rectangular structuring element is used. Kernel can be created using #getStructuringElementanchor - position of the anchor within the element; default value (-1, -1) means that the anchor is at the element center. SEE: erode, morphologyEx, getStructuringElement

#### dilate public static void dilate([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int iterations) Dilates an image by using a specific structuring element. The function dilates the source image using the specified structuring element that determines the shape of a pixel neighborhood over which the maximum is taken: \(\texttt{dst} (x,y) = \max \_{(x',y'): \, \texttt{element} (x',y') \ne0 } \texttt{src} (x+x',y+y')\) The function supports the in-place mode. Dilation can be applied several ( iterations ) times. In case of multi-channel images, each channel is processed independently.Parameters:src - input image; the number of channels can be arbitrary, but the depth should be one of CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - output image of the same size and type as src.kernel - structuring element used for dilation; if elemenat=Mat(), a 3 x 3 rectangular structuring element is used. Kernel can be created using #getStructuringElementanchor - position of the anchor within the element; default value (-1, -1) means that the anchor is at the element center.iterations - number of times dilation is applied. SEE: erode, morphologyEx, getStructuringElement

#### dilate public static void dilate([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int iterations, int borderType) Dilates an image by using a specific structuring element. The function dilates the source image using the specified structuring element that determines the shape of a pixel neighborhood over which the maximum is taken: \(\texttt{dst} (x,y) = \max \_{(x',y'): \, \texttt{element} (x',y') \ne0 } \texttt{src} (x+x',y+y')\) The function supports the in-place mode. Dilation can be applied several ( iterations ) times. In case of multi-channel images, each channel is processed independently.Parameters:src - input image; the number of channels can be arbitrary, but the depth should be one of CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - output image of the same size and type as src.kernel - structuring element used for dilation; if elemenat=Mat(), a 3 x 3 rectangular structuring element is used. Kernel can be created using #getStructuringElementanchor - position of the anchor within the element; default value (-1, -1) means that the anchor is at the element center.iterations - number of times dilation is applied.borderType - pixel extrapolation method, see #BorderTypes. #BORDER\_WRAP is not suported. SEE: erode, morphologyEx, getStructuringElement

#### dilate public static void dilate([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int iterations, int borderType, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) borderValue) Dilates an image by using a specific structuring element. The function dilates the source image using the specified structuring element that determines the shape of a pixel neighborhood over which the maximum is taken: \(\texttt{dst} (x,y) = \max \_{(x',y'): \, \texttt{element} (x',y') \ne0 } \texttt{src} (x+x',y+y')\) The function supports the in-place mode. Dilation can be applied several ( iterations ) times. In case of multi-channel images, each channel is processed independently.Parameters:src - input image; the number of channels can be arbitrary, but the depth should be one of CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - output image of the same size and type as src.kernel - structuring element used for dilation; if elemenat=Mat(), a 3 x 3 rectangular structuring element is used. Kernel can be created using #getStructuringElementanchor - position of the anchor within the element; default value (-1, -1) means that the anchor is at the element center.iterations - number of times dilation is applied.borderType - pixel extrapolation method, see #BorderTypes. #BORDER\_WRAP is not suported.borderValue - border value in case of a constant border SEE: erode, morphologyEx, getStructuringElement

#### distanceTransform public static void distanceTransform([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int distanceType, int maskSize)Parameters:src - 8-bit, single-channel (binary) source image.dst - Output image with calculated distances. It is a 8-bit or 32-bit floating-point, single-channel image of the same size as src .distanceType - Type of distance, see #DistanceTypesmaskSize - Size of the distance transform mask, see #DistanceTransformMasks. In case of the #DIST\_L1 or #DIST\_C distance type, the parameter is forced to 3 because a \(3\times 3\) mask gives the same result as \(5\times 5\) or any larger aperture. the first variant of the function and distanceType == #DIST\_L1.

#### distanceTransform public static void distanceTransform([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int distanceType, int maskSize, int dstType)Parameters:src - 8-bit, single-channel (binary) source image.dst - Output image with calculated distances. It is a 8-bit or 32-bit floating-point, single-channel image of the same size as src .distanceType - Type of distance, see #DistanceTypesmaskSize - Size of the distance transform mask, see #DistanceTransformMasks. In case of the #DIST\_L1 or #DIST\_C distance type, the parameter is forced to 3 because a \(3\times 3\) mask gives the same result as \(5\times 5\) or any larger aperture.dstType - Type of output image. It can be CV\_8U or CV\_32F. Type CV\_8U can be used only for the first variant of the function and distanceType == #DIST\_L1.

#### distanceTransformWithLabels public static void distanceTransformWithLabels([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels, int distanceType, int maskSize)

Calculates the distance to the closest zero pixel for each pixel of the source image. The function cv::distanceTransform calculates the approximate or precise distance from every binary image pixel to the nearest zero pixel. For zero image pixels, the distance will obviously be zero. When maskSize == #DIST\_MASK\_PRECISE and distanceType == #DIST\_L2 , the function runs the algorithm described in CITE: Felzenszwalb04 . This algorithm is parallelized with the TBB library. In other cases, the algorithm CITE: Borgefors86 is used. This means that for a pixel the function finds the shortest path to the nearest zero pixel consisting of basic shifts: horizontal, vertical, diagonal, or knight's move (the latest is available for a \(5\times 5\) mask). The overall distance is calculated as a sum of these basic distances. Since the distance function should be symmetric, all of the horizontal and vertical shifts must have the same cost (denoted as a ), all the diagonal shifts must have the same cost (denoted as b), and all knight's moves must have the same cost (denoted as c). For the #DIST\_C and #DIST\_L1 types, the distance is calculated precisely, whereas for #DIST\_L2 (Euclidean distance) the distance can be calculated only with a relative error (a \(5\times 5\) mask gives more accurate results). For a,b, and c, OpenCV uses the values suggested in the original paper:

* + - * DIST\_L1: a = 1, b = 2
      * DIST\_L2:
        + 3 x 3: a=0.955, b=1.3693
        + 5 x 5: a=1, b=1.4, c=2.1969
      * DIST\_C: a = 1, b = 1

Typically, for a fast, coarse distance estimation #DIST\_L2, a \(3\times 3\) mask is used. For a more accurate distance estimation #DIST\_L2, a \(5\times 5\) mask or the precise algorithm is used. Note that both the precise and the approximate algorithms are linear on the number of pixels. This variant of the function does not only compute the minimum distance for each pixel \((x, y)\) but also identifies the nearest connected component consisting of zero pixels (labelType==#DIST\_LABEL\_CCOMP) or the nearest zero pixel (labelType==#DIST\_LABEL\_PIXEL). Index of the component/pixel is stored in labels(x, y). When labelType==#DIST\_LABEL\_CCOMP, the function automatically finds connected components of zero pixels in the input image and marks them with distinct labels. When labelType==#DIST\_LABEL\_PIXEL, the function scans through the input image and marks all the zero pixels with distinct labels. In this mode, the complexity is still linear. That is, the function provides a very fast way to compute the Voronoi diagram for a binary image. Currently, the second variant can use only the approximate distance transform algorithm, i.e. maskSize=#DIST\_MASK\_PRECISE is not supported yet.Parameters:src - 8-bit, single-channel (binary) source image.dst - Output image with calculated distances. It is a 8-bit or 32-bit floating-point, single-channel image of the same size as src.labels - Output 2D array of labels (the discrete Voronoi diagram). It has the type CV\_32SC1 and the same size as src.distanceType - Type of distance, see #DistanceTypesmaskSize - Size of the distance transform mask, see #DistanceTransformMasks. #DIST\_MASK\_PRECISE is not supported by this variant. In case of the #DIST\_L1 or #DIST\_C distance type, the parameter is forced to 3 because a \(3\times 3\) mask gives the same result as \(5\times 5\) or any larger aperture.

#### distanceTransformWithLabels public static void distanceTransformWithLabels([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) labels, int distanceType, int maskSize, int labelType)

Calculates the distance to the closest zero pixel for each pixel of the source image. The function cv::distanceTransform calculates the approximate or precise distance from every binary image pixel to the nearest zero pixel. For zero image pixels, the distance will obviously be zero. When maskSize == #DIST\_MASK\_PRECISE and distanceType == #DIST\_L2 , the function runs the algorithm described in CITE: Felzenszwalb04 . This algorithm is parallelized with the TBB library. In other cases, the algorithm CITE: Borgefors86 is used. This means that for a pixel the function finds the shortest path to the nearest zero pixel consisting of basic shifts: horizontal, vertical, diagonal, or knight's move (the latest is available for a \(5\times 5\) mask). The overall distance is calculated as a sum of these basic distances. Since the distance function should be symmetric, all of the horizontal and vertical shifts must have the same cost (denoted as a ), all the diagonal shifts must have the same cost (denoted as b), and all knight's moves must have the same cost (denoted as c). For the #DIST\_C and #DIST\_L1 types, the distance is calculated precisely, whereas for #DIST\_L2 (Euclidean distance) the distance can be calculated only with a relative error (a \(5\times 5\) mask gives more accurate results). For a,b, and c, OpenCV uses the values suggested in the original paper:

* + - * DIST\_L1: a = 1, b = 2
      * DIST\_L2:
        + 3 x 3: a=0.955, b=1.3693
        + 5 x 5: a=1, b=1.4, c=2.1969
      * DIST\_C: a = 1, b = 1

Typically, for a fast, coarse distance estimation #DIST\_L2, a \(3\times 3\) mask is used. For a more accurate distance estimation #DIST\_L2, a \(5\times 5\) mask or the precise algorithm is used. Note that both the precise and the approximate algorithms are linear on the number of pixels. This variant of the function does not only compute the minimum distance for each pixel \((x, y)\) but also identifies the nearest connected component consisting of zero pixels (labelType==#DIST\_LABEL\_CCOMP) or the nearest zero pixel (labelType==#DIST\_LABEL\_PIXEL). Index of the component/pixel is stored in labels(x, y). When labelType==#DIST\_LABEL\_CCOMP, the function automatically finds connected components of zero pixels in the input image and marks them with distinct labels. When labelType==#DIST\_LABEL\_PIXEL, the function scans through the input image and marks all the zero pixels with distinct labels. In this mode, the complexity is still linear. That is, the function provides a very fast way to compute the Voronoi diagram for a binary image. Currently, the second variant can use only the approximate distance transform algorithm, i.e. maskSize=#DIST\_MASK\_PRECISE is not supported yet.Parameters:src - 8-bit, single-channel (binary) source image.dst - Output image with calculated distances. It is a 8-bit or 32-bit floating-point, single-channel image of the same size as src.labels - Output 2D array of labels (the discrete Voronoi diagram). It has the type CV\_32SC1 and the same size as src.distanceType - Type of distance, see #DistanceTypesmaskSize - Size of the distance transform mask, see #DistanceTransformMasks. #DIST\_MASK\_PRECISE is not supported by this variant. In case of the #DIST\_L1 or #DIST\_C distance type, the parameter is forced to 3 because a \(3\times 3\) mask gives the same result as \(5\times 5\) or any larger aperture.labelType - Type of the label array to build, see #DistanceTransformLabelTypes.

#### drawContours public static void drawContours([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> contours, int contourIdx, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Draws contours outlines or filled contours. The function draws contour outlines in the image if \(\texttt{thickness} \ge 0\) or fills the area bounded by the contours if \(\texttt{thickness}<0\) . The example below shows how to retrieve connected components from the binary image and label them: : INCLUDE: snippets/imgproc\_drawContours.cppParameters:image - Destination image.contours - All the input contours. Each contour is stored as a point vector.contourIdx - Parameter indicating a contour to draw. If it is negative, all the contours are drawn.color - Color of the contours. thickness=#FILLED ), the contour interiors are drawn. some of the contours (see maxLevel ). If it is 1, the function draws the contour(s) and all the nested contours. If it is 2, the function draws the contours, all the nested contours, all the nested-to-nested contours, and so on. This parameter is only taken into account when there is hierarchy available. \(\texttt{offset}=(dx,dy)\) . **Note:** When thickness=#FILLED, the function is designed to handle connected components with holes correctly even when no hierarchy date is provided. This is done by analyzing all the outlines together using even-odd rule. This may give incorrect results if you have a joint collection of separately retrieved contours. In order to solve this problem, you need to call #drawContours separately for each sub-group of contours, or iterate over the collection using contourIdx parameter.

#### drawContours public static void drawContours([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> contours, int contourIdx, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness) Draws contours outlines or filled contours. The function draws contour outlines in the image if \(\texttt{thickness} \ge 0\) or fills the area bounded by the contours if \(\texttt{thickness}<0\) . The example below shows how to retrieve connected components from the binary image and label them: : INCLUDE: snippets/imgproc\_drawContours.cppParameters:image - Destination image.contours - All the input contours. Each contour is stored as a point vector.contourIdx - Parameter indicating a contour to draw. If it is negative, all the contours are drawn.color - Color of the contours.thickness - Thickness of lines the contours are drawn with. If it is negative (for example, thickness=#FILLED ), the contour interiors are drawn. some of the contours (see maxLevel ). If it is 1, the function draws the contour(s) and all the nested contours. If it is 2, the function draws the contours, all the nested contours, all the nested-to-nested contours, and so on. This parameter is only taken into account when there is hierarchy available. \(\texttt{offset}=(dx,dy)\) . **Note:** When thickness=#FILLED, the function is designed to handle connected components with holes correctly even when no hierarchy date is provided. This is done by analyzing all the outlines together using even-odd rule. This may give incorrect results if you have a joint collection of separately retrieved contours. In order to solve this problem, you need to call #drawContours separately for each sub-group of contours, or iterate over the collection using contourIdx parameter.

#### drawContours public static void drawContours([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> contours, int contourIdx, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType) Draws contours outlines or filled contours. The function draws contour outlines in the image if \(\texttt{thickness} \ge 0\) or fills the area bounded by the contours if \(\texttt{thickness}<0\) . The example below shows how to retrieve connected components from the binary image and label them: : INCLUDE: snippets/imgproc\_drawContours.cppParameters:image - Destination image.contours - All the input contours. Each contour is stored as a point vector.contourIdx - Parameter indicating a contour to draw. If it is negative, all the contours are drawn.color - Color of the contours.thickness - Thickness of lines the contours are drawn with. If it is negative (for example, thickness=#FILLED ), the contour interiors are drawn.lineType - Line connectivity. See #LineTypes some of the contours (see maxLevel ). If it is 1, the function draws the contour(s) and all the nested contours. If it is 2, the function draws the contours, all the nested contours, all the nested-to-nested contours, and so on. This parameter is only taken into account when there is hierarchy available. \(\texttt{offset}=(dx,dy)\) . **Note:** When thickness=#FILLED, the function is designed to handle connected components with holes correctly even when no hierarchy date is provided. This is done by analyzing all the outlines together using even-odd rule. This may give incorrect results if you have a joint collection of separately retrieved contours. In order to solve this problem, you need to call #drawContours separately for each sub-group of contours, or iterate over the collection using contourIdx parameter.

#### drawContours public static void drawContours([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> contours, int contourIdx, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType, [Mat](http://docs.google.com/org/opencv/core/Mat.html) hierarchy) Draws contours outlines or filled contours. The function draws contour outlines in the image if \(\texttt{thickness} \ge 0\) or fills the area bounded by the contours if \(\texttt{thickness}<0\) . The example below shows how to retrieve connected components from the binary image and label them: : INCLUDE: snippets/imgproc\_drawContours.cppParameters:image - Destination image.contours - All the input contours. Each contour is stored as a point vector.contourIdx - Parameter indicating a contour to draw. If it is negative, all the contours are drawn.color - Color of the contours.thickness - Thickness of lines the contours are drawn with. If it is negative (for example, thickness=#FILLED ), the contour interiors are drawn.lineType - Line connectivity. See #LineTypeshierarchy - Optional information about hierarchy. It is only needed if you want to draw only some of the contours (see maxLevel ). If it is 1, the function draws the contour(s) and all the nested contours. If it is 2, the function draws the contours, all the nested contours, all the nested-to-nested contours, and so on. This parameter is only taken into account when there is hierarchy available. \(\texttt{offset}=(dx,dy)\) . **Note:** When thickness=#FILLED, the function is designed to handle connected components with holes correctly even when no hierarchy date is provided. This is done by analyzing all the outlines together using even-odd rule. This may give incorrect results if you have a joint collection of separately retrieved contours. In order to solve this problem, you need to call #drawContours separately for each sub-group of contours, or iterate over the collection using contourIdx parameter.

#### drawContours public static void drawContours([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> contours, int contourIdx, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType, [Mat](http://docs.google.com/org/opencv/core/Mat.html) hierarchy, int maxLevel) Draws contours outlines or filled contours. The function draws contour outlines in the image if \(\texttt{thickness} \ge 0\) or fills the area bounded by the contours if \(\texttt{thickness}<0\) . The example below shows how to retrieve connected components from the binary image and label them: : INCLUDE: snippets/imgproc\_drawContours.cppParameters:image - Destination image.contours - All the input contours. Each contour is stored as a point vector.contourIdx - Parameter indicating a contour to draw. If it is negative, all the contours are drawn.color - Color of the contours.thickness - Thickness of lines the contours are drawn with. If it is negative (for example, thickness=#FILLED ), the contour interiors are drawn.lineType - Line connectivity. See #LineTypeshierarchy - Optional information about hierarchy. It is only needed if you want to draw only some of the contours (see maxLevel ).maxLevel - Maximal level for drawn contours. If it is 0, only the specified contour is drawn. If it is 1, the function draws the contour(s) and all the nested contours. If it is 2, the function draws the contours, all the nested contours, all the nested-to-nested contours, and so on. This parameter is only taken into account when there is hierarchy available. \(\texttt{offset}=(dx,dy)\) . **Note:** When thickness=#FILLED, the function is designed to handle connected components with holes correctly even when no hierarchy date is provided. This is done by analyzing all the outlines together using even-odd rule. This may give incorrect results if you have a joint collection of separately retrieved contours. In order to solve this problem, you need to call #drawContours separately for each sub-group of contours, or iterate over the collection using contourIdx parameter.

#### drawContours public static void drawContours([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> contours, int contourIdx, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType, [Mat](http://docs.google.com/org/opencv/core/Mat.html) hierarchy, int maxLevel, [Point](http://docs.google.com/org/opencv/core/Point.html) offset) Draws contours outlines or filled contours. The function draws contour outlines in the image if \(\texttt{thickness} \ge 0\) or fills the area bounded by the contours if \(\texttt{thickness}<0\) . The example below shows how to retrieve connected components from the binary image and label them: : INCLUDE: snippets/imgproc\_drawContours.cppParameters:image - Destination image.contours - All the input contours. Each contour is stored as a point vector.contourIdx - Parameter indicating a contour to draw. If it is negative, all the contours are drawn.color - Color of the contours.thickness - Thickness of lines the contours are drawn with. If it is negative (for example, thickness=#FILLED ), the contour interiors are drawn.lineType - Line connectivity. See #LineTypeshierarchy - Optional information about hierarchy. It is only needed if you want to draw only some of the contours (see maxLevel ).maxLevel - Maximal level for drawn contours. If it is 0, only the specified contour is drawn. If it is 1, the function draws the contour(s) and all the nested contours. If it is 2, the function draws the contours, all the nested contours, all the nested-to-nested contours, and so on. This parameter is only taken into account when there is hierarchy available.offset - Optional contour shift parameter. Shift all the drawn contours by the specified \(\texttt{offset}=(dx,dy)\) . **Note:** When thickness=#FILLED, the function is designed to handle connected components with holes correctly even when no hierarchy date is provided. This is done by analyzing all the outlines together using even-odd rule. This may give incorrect results if you have a joint collection of separately retrieved contours. In order to solve this problem, you need to call #drawContours separately for each sub-group of contours, or iterate over the collection using contourIdx parameter.

#### drawMarker public static void drawMarker([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) position, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Draws a marker on a predefined position in an image. The function cv::drawMarker draws a marker on a given position in the image. For the moment several marker types are supported, see #MarkerTypes for more information.Parameters:img - Image.position - The point where the crosshair is positioned.color - Line color.

#### drawMarker public static void drawMarker([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) position, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int markerType) Draws a marker on a predefined position in an image. The function cv::drawMarker draws a marker on a given position in the image. For the moment several marker types are supported, see #MarkerTypes for more information.Parameters:img - Image.position - The point where the crosshair is positioned.color - Line color.markerType - The specific type of marker you want to use, see #MarkerTypes

#### drawMarker public static void drawMarker([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) position, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int markerType, int markerSize) Draws a marker on a predefined position in an image. The function cv::drawMarker draws a marker on a given position in the image. For the moment several marker types are supported, see #MarkerTypes for more information.Parameters:img - Image.position - The point where the crosshair is positioned.color - Line color.markerType - The specific type of marker you want to use, see #MarkerTypesmarkerSize - The length of the marker axis [default = 20 pixels]

#### drawMarker public static void drawMarker([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) position, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int markerType, int markerSize, int thickness) Draws a marker on a predefined position in an image. The function cv::drawMarker draws a marker on a given position in the image. For the moment several marker types are supported, see #MarkerTypes for more information.Parameters:img - Image.position - The point where the crosshair is positioned.color - Line color.markerType - The specific type of marker you want to use, see #MarkerTypesthickness - Line thickness.markerSize - The length of the marker axis [default = 20 pixels]

#### drawMarker public static void drawMarker([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) position, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int markerType, int markerSize, int thickness, int line\_type) Draws a marker on a predefined position in an image. The function cv::drawMarker draws a marker on a given position in the image. For the moment several marker types are supported, see #MarkerTypes for more information.Parameters:img - Image.position - The point where the crosshair is positioned.color - Line color.markerType - The specific type of marker you want to use, see #MarkerTypesthickness - Line thickness.line\_type - Type of the line, See #LineTypesmarkerSize - The length of the marker axis [default = 20 pixels]

#### ellipse public static void ellipse([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) center, [Size](http://docs.google.com/org/opencv/core/Size.html) axes, double angle, double startAngle, double endAngle, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Draws a simple or thick elliptic arc or fills an ellipse sector. The function cv::ellipse with more parameters draws an ellipse outline, a filled ellipse, an elliptic arc, or a filled ellipse sector. The drawing code uses general parametric form. A piecewise-linear curve is used to approximate the elliptic arc boundary. If you need more control of the ellipse rendering, you can retrieve the curve using #ellipse2Poly and then render it with #polylines or fill it with #fillPoly. If you use the first variant of the function and want to draw the whole ellipse, not an arc, pass startAngle=0 and endAngle=360. If startAngle is greater than endAngle, they are swapped. The figure below explains the meaning of the parameters to draw the blue arc. ![Parameters of Elliptic Arc](pics/ellipse.svg)Parameters:img - Image.center - Center of the ellipse.axes - Half of the size of the ellipse main axes.angle - Ellipse rotation angle in degrees.startAngle - Starting angle of the elliptic arc in degrees.endAngle - Ending angle of the elliptic arc in degrees.color - Ellipse color. a filled ellipse sector is to be drawn.

#### ellipse public static void ellipse([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) center, [Size](http://docs.google.com/org/opencv/core/Size.html) axes, double angle, double startAngle, double endAngle, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness) Draws a simple or thick elliptic arc or fills an ellipse sector. The function cv::ellipse with more parameters draws an ellipse outline, a filled ellipse, an elliptic arc, or a filled ellipse sector. The drawing code uses general parametric form. A piecewise-linear curve is used to approximate the elliptic arc boundary. If you need more control of the ellipse rendering, you can retrieve the curve using #ellipse2Poly and then render it with #polylines or fill it with #fillPoly. If you use the first variant of the function and want to draw the whole ellipse, not an arc, pass startAngle=0 and endAngle=360. If startAngle is greater than endAngle, they are swapped. The figure below explains the meaning of the parameters to draw the blue arc. ![Parameters of Elliptic Arc](pics/ellipse.svg)Parameters:img - Image.center - Center of the ellipse.axes - Half of the size of the ellipse main axes.angle - Ellipse rotation angle in degrees.startAngle - Starting angle of the elliptic arc in degrees.endAngle - Ending angle of the elliptic arc in degrees.color - Ellipse color.thickness - Thickness of the ellipse arc outline, if positive. Otherwise, this indicates that a filled ellipse sector is to be drawn.

#### ellipse public static void ellipse([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) center, [Size](http://docs.google.com/org/opencv/core/Size.html) axes, double angle, double startAngle, double endAngle, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType) Draws a simple or thick elliptic arc or fills an ellipse sector. The function cv::ellipse with more parameters draws an ellipse outline, a filled ellipse, an elliptic arc, or a filled ellipse sector. The drawing code uses general parametric form. A piecewise-linear curve is used to approximate the elliptic arc boundary. If you need more control of the ellipse rendering, you can retrieve the curve using #ellipse2Poly and then render it with #polylines or fill it with #fillPoly. If you use the first variant of the function and want to draw the whole ellipse, not an arc, pass startAngle=0 and endAngle=360. If startAngle is greater than endAngle, they are swapped. The figure below explains the meaning of the parameters to draw the blue arc. ![Parameters of Elliptic Arc](pics/ellipse.svg)Parameters:img - Image.center - Center of the ellipse.axes - Half of the size of the ellipse main axes.angle - Ellipse rotation angle in degrees.startAngle - Starting angle of the elliptic arc in degrees.endAngle - Ending angle of the elliptic arc in degrees.color - Ellipse color.thickness - Thickness of the ellipse arc outline, if positive. Otherwise, this indicates that a filled ellipse sector is to be drawn.lineType - Type of the ellipse boundary. See #LineTypes

#### ellipse public static void ellipse([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) center, [Size](http://docs.google.com/org/opencv/core/Size.html) axes, double angle, double startAngle, double endAngle, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType, int shift) Draws a simple or thick elliptic arc or fills an ellipse sector. The function cv::ellipse with more parameters draws an ellipse outline, a filled ellipse, an elliptic arc, or a filled ellipse sector. The drawing code uses general parametric form. A piecewise-linear curve is used to approximate the elliptic arc boundary. If you need more control of the ellipse rendering, you can retrieve the curve using #ellipse2Poly and then render it with #polylines or fill it with #fillPoly. If you use the first variant of the function and want to draw the whole ellipse, not an arc, pass startAngle=0 and endAngle=360. If startAngle is greater than endAngle, they are swapped. The figure below explains the meaning of the parameters to draw the blue arc. ![Parameters of Elliptic Arc](pics/ellipse.svg)Parameters:img - Image.center - Center of the ellipse.axes - Half of the size of the ellipse main axes.angle - Ellipse rotation angle in degrees.startAngle - Starting angle of the elliptic arc in degrees.endAngle - Ending angle of the elliptic arc in degrees.color - Ellipse color.thickness - Thickness of the ellipse arc outline, if positive. Otherwise, this indicates that a filled ellipse sector is to be drawn.lineType - Type of the ellipse boundary. See #LineTypesshift - Number of fractional bits in the coordinates of the center and values of axes.

#### ellipse public static void ellipse([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) box, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color)Parameters:img - Image.box - Alternative ellipse representation via RotatedRect. This means that the function draws an ellipse inscribed in the rotated rectangle.color - Ellipse color. a filled ellipse sector is to be drawn.

#### ellipse public static void ellipse([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) box, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness)Parameters:img - Image.box - Alternative ellipse representation via RotatedRect. This means that the function draws an ellipse inscribed in the rotated rectangle.color - Ellipse color.thickness - Thickness of the ellipse arc outline, if positive. Otherwise, this indicates that a filled ellipse sector is to be drawn.

#### ellipse public static void ellipse([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) box, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType)Parameters:img - Image.box - Alternative ellipse representation via RotatedRect. This means that the function draws an ellipse inscribed in the rotated rectangle.color - Ellipse color.thickness - Thickness of the ellipse arc outline, if positive. Otherwise, this indicates that a filled ellipse sector is to be drawn.lineType - Type of the ellipse boundary. See #LineTypes

#### ellipse2Poly public static void ellipse2Poly([Point](http://docs.google.com/org/opencv/core/Point.html) center, [Size](http://docs.google.com/org/opencv/core/Size.html) axes, int angle, int arcStart, int arcEnd, int delta, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) pts) Approximates an elliptic arc with a polyline. The function ellipse2Poly computes the vertices of a polyline that approximates the specified elliptic arc. It is used by #ellipse. If arcStart is greater than arcEnd, they are swapped.Parameters:center - Center of the arc.axes - Half of the size of the ellipse main axes. See #ellipse for details.angle - Rotation angle of the ellipse in degrees. See #ellipse for details.arcStart - Starting angle of the elliptic arc in degrees.arcEnd - Ending angle of the elliptic arc in degrees.delta - Angle between the subsequent polyline vertices. It defines the approximation accuracy.pts - Output vector of polyline vertices.

#### EMD public static float EMD([Mat](http://docs.google.com/org/opencv/core/Mat.html) signature1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) signature2, int distType) Computes the "minimal work" distance between two weighted point configurations. The function computes the earth mover distance and/or a lower boundary of the distance between the two weighted point configurations. One of the applications described in CITE: RubnerSept98, CITE: Rubner2000 is multi-dimensional histogram comparison for image retrieval. EMD is a transportation problem that is solved using some modification of a simplex algorithm, thus the complexity is exponential in the worst case, though, on average it is much faster. In the case of a real metric the lower boundary can be calculated even faster (using linear-time algorithm) and it can be used to determine roughly whether the two signatures are far enough so that they cannot relate to the same object.Parameters:signature1 - First signature, a \(\texttt{size1}\times \texttt{dims}+1\) floating-point matrix. Each row stores the point weight followed by the point coordinates. The matrix is allowed to have a single column (weights only) if the user-defined cost matrix is used. The weights must be non-negative and have at least one non-zero value.signature2 - Second signature of the same format as signature1 , though the number of rows may be different. The total weights may be different. In this case an extra "dummy" point is added to either signature1 or signature2. The weights must be non-negative and have at least one non-zero value.distType - Used metric. See #DistanceTypes. is used, lower boundary lowerBound cannot be calculated because it needs a metric function. signatures that is a distance between mass centers. The lower boundary may not be calculated if the user-defined cost matrix is used, the total weights of point configurations are not equal, or if the signatures consist of weights only (the signature matrices have a single column). You **must** initialize \\*lowerBound . If the calculated distance between mass centers is greater or equal to \\*lowerBound (it means that the signatures are far enough), the function does not calculate EMD. In any case \\*lowerBound is set to the calculated distance between mass centers on return. Thus, if you want to calculate both distance between mass centers and EMD, \\*lowerBound should be set to 0. a flow from \(i\) -th point of signature1 to \(j\) -th point of signature2 . Returns:automatically generated

#### EMD public static float EMD([Mat](http://docs.google.com/org/opencv/core/Mat.html) signature1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) signature2, int distType, [Mat](http://docs.google.com/org/opencv/core/Mat.html) cost) Computes the "minimal work" distance between two weighted point configurations. The function computes the earth mover distance and/or a lower boundary of the distance between the two weighted point configurations. One of the applications described in CITE: RubnerSept98, CITE: Rubner2000 is multi-dimensional histogram comparison for image retrieval. EMD is a transportation problem that is solved using some modification of a simplex algorithm, thus the complexity is exponential in the worst case, though, on average it is much faster. In the case of a real metric the lower boundary can be calculated even faster (using linear-time algorithm) and it can be used to determine roughly whether the two signatures are far enough so that they cannot relate to the same object.Parameters:signature1 - First signature, a \(\texttt{size1}\times \texttt{dims}+1\) floating-point matrix. Each row stores the point weight followed by the point coordinates. The matrix is allowed to have a single column (weights only) if the user-defined cost matrix is used. The weights must be non-negative and have at least one non-zero value.signature2 - Second signature of the same format as signature1 , though the number of rows may be different. The total weights may be different. In this case an extra "dummy" point is added to either signature1 or signature2. The weights must be non-negative and have at least one non-zero value.distType - Used metric. See #DistanceTypes.cost - User-defined \(\texttt{size1}\times \texttt{size2}\) cost matrix. Also, if a cost matrix is used, lower boundary lowerBound cannot be calculated because it needs a metric function. signatures that is a distance between mass centers. The lower boundary may not be calculated if the user-defined cost matrix is used, the total weights of point configurations are not equal, or if the signatures consist of weights only (the signature matrices have a single column). You **must** initialize \\*lowerBound . If the calculated distance between mass centers is greater or equal to \\*lowerBound (it means that the signatures are far enough), the function does not calculate EMD. In any case \\*lowerBound is set to the calculated distance between mass centers on return. Thus, if you want to calculate both distance between mass centers and EMD, \\*lowerBound should be set to 0. a flow from \(i\) -th point of signature1 to \(j\) -th point of signature2 . Returns:automatically generated

#### EMD public static float EMD([Mat](http://docs.google.com/org/opencv/core/Mat.html) signature1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) signature2, int distType, [Mat](http://docs.google.com/org/opencv/core/Mat.html) cost, [Mat](http://docs.google.com/org/opencv/core/Mat.html) flow) Computes the "minimal work" distance between two weighted point configurations. The function computes the earth mover distance and/or a lower boundary of the distance between the two weighted point configurations. One of the applications described in CITE: RubnerSept98, CITE: Rubner2000 is multi-dimensional histogram comparison for image retrieval. EMD is a transportation problem that is solved using some modification of a simplex algorithm, thus the complexity is exponential in the worst case, though, on average it is much faster. In the case of a real metric the lower boundary can be calculated even faster (using linear-time algorithm) and it can be used to determine roughly whether the two signatures are far enough so that they cannot relate to the same object.Parameters:signature1 - First signature, a \(\texttt{size1}\times \texttt{dims}+1\) floating-point matrix. Each row stores the point weight followed by the point coordinates. The matrix is allowed to have a single column (weights only) if the user-defined cost matrix is used. The weights must be non-negative and have at least one non-zero value.signature2 - Second signature of the same format as signature1 , though the number of rows may be different. The total weights may be different. In this case an extra "dummy" point is added to either signature1 or signature2. The weights must be non-negative and have at least one non-zero value.distType - Used metric. See #DistanceTypes.cost - User-defined \(\texttt{size1}\times \texttt{size2}\) cost matrix. Also, if a cost matrix is used, lower boundary lowerBound cannot be calculated because it needs a metric function. signatures that is a distance between mass centers. The lower boundary may not be calculated if the user-defined cost matrix is used, the total weights of point configurations are not equal, or if the signatures consist of weights only (the signature matrices have a single column). You **must** initialize \\*lowerBound . If the calculated distance between mass centers is greater or equal to \\*lowerBound (it means that the signatures are far enough), the function does not calculate EMD. In any case \\*lowerBound is set to the calculated distance between mass centers on return. Thus, if you want to calculate both distance between mass centers and EMD, \\*lowerBound should be set to 0.flow - Resultant \(\texttt{size1} \times \texttt{size2}\) flow matrix: \(\texttt{flow}\_{i,j}\) is a flow from \(i\) -th point of signature1 to \(j\) -th point of signature2 . Returns:automatically generated

#### equalizeHist public static void equalizeHist([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst)

Equalizes the histogram of a grayscale image. The function equalizes the histogram of the input image using the following algorithm:

* + - * Calculate the histogram \(H\) for src .
      * Normalize the histogram so that the sum of histogram bins is 255.
      * Compute the integral of the histogram: \(H'\_i = \sum \_{0 \le j < i} H(j)\)
      * Transform the image using \(H'\) as a look-up table: \(\texttt{dst}(x,y) = H'(\texttt{src}(x,y))\)

The algorithm normalizes the brightness and increases the contrast of the image.Parameters:src - Source 8-bit single channel image.dst - Destination image of the same size and type as src .

#### erode public static void erode([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel) Erodes an image by using a specific structuring element. The function erodes the source image using the specified structuring element that determines the shape of a pixel neighborhood over which the minimum is taken: \(\texttt{dst} (x,y) = \min \_{(x',y'): \, \texttt{element} (x',y') \ne0 } \texttt{src} (x+x',y+y')\) The function supports the in-place mode. Erosion can be applied several ( iterations ) times. In case of multi-channel images, each channel is processed independently.Parameters:src - input image; the number of channels can be arbitrary, but the depth should be one of CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - output image of the same size and type as src.kernel - structuring element used for erosion; if element=Mat(), a 3 x 3 rectangular structuring element is used. Kernel can be created using #getStructuringElement. anchor is at the element center. SEE: dilate, morphologyEx, getStructuringElement

#### erode public static void erode([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor) Erodes an image by using a specific structuring element. The function erodes the source image using the specified structuring element that determines the shape of a pixel neighborhood over which the minimum is taken: \(\texttt{dst} (x,y) = \min \_{(x',y'): \, \texttt{element} (x',y') \ne0 } \texttt{src} (x+x',y+y')\) The function supports the in-place mode. Erosion can be applied several ( iterations ) times. In case of multi-channel images, each channel is processed independently.Parameters:src - input image; the number of channels can be arbitrary, but the depth should be one of CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - output image of the same size and type as src.kernel - structuring element used for erosion; if element=Mat(), a 3 x 3 rectangular structuring element is used. Kernel can be created using #getStructuringElement.anchor - position of the anchor within the element; default value (-1, -1) means that the anchor is at the element center. SEE: dilate, morphologyEx, getStructuringElement

#### erode public static void erode([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int iterations) Erodes an image by using a specific structuring element. The function erodes the source image using the specified structuring element that determines the shape of a pixel neighborhood over which the minimum is taken: \(\texttt{dst} (x,y) = \min \_{(x',y'): \, \texttt{element} (x',y') \ne0 } \texttt{src} (x+x',y+y')\) The function supports the in-place mode. Erosion can be applied several ( iterations ) times. In case of multi-channel images, each channel is processed independently.Parameters:src - input image; the number of channels can be arbitrary, but the depth should be one of CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - output image of the same size and type as src.kernel - structuring element used for erosion; if element=Mat(), a 3 x 3 rectangular structuring element is used. Kernel can be created using #getStructuringElement.anchor - position of the anchor within the element; default value (-1, -1) means that the anchor is at the element center.iterations - number of times erosion is applied. SEE: dilate, morphologyEx, getStructuringElement

#### erode public static void erode([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int iterations, int borderType) Erodes an image by using a specific structuring element. The function erodes the source image using the specified structuring element that determines the shape of a pixel neighborhood over which the minimum is taken: \(\texttt{dst} (x,y) = \min \_{(x',y'): \, \texttt{element} (x',y') \ne0 } \texttt{src} (x+x',y+y')\) The function supports the in-place mode. Erosion can be applied several ( iterations ) times. In case of multi-channel images, each channel is processed independently.Parameters:src - input image; the number of channels can be arbitrary, but the depth should be one of CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - output image of the same size and type as src.kernel - structuring element used for erosion; if element=Mat(), a 3 x 3 rectangular structuring element is used. Kernel can be created using #getStructuringElement.anchor - position of the anchor within the element; default value (-1, -1) means that the anchor is at the element center.iterations - number of times erosion is applied.borderType - pixel extrapolation method, see #BorderTypes. #BORDER\_WRAP is not supported. SEE: dilate, morphologyEx, getStructuringElement

#### erode public static void erode([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int iterations, int borderType, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) borderValue) Erodes an image by using a specific structuring element. The function erodes the source image using the specified structuring element that determines the shape of a pixel neighborhood over which the minimum is taken: \(\texttt{dst} (x,y) = \min \_{(x',y'): \, \texttt{element} (x',y') \ne0 } \texttt{src} (x+x',y+y')\) The function supports the in-place mode. Erosion can be applied several ( iterations ) times. In case of multi-channel images, each channel is processed independently.Parameters:src - input image; the number of channels can be arbitrary, but the depth should be one of CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - output image of the same size and type as src.kernel - structuring element used for erosion; if element=Mat(), a 3 x 3 rectangular structuring element is used. Kernel can be created using #getStructuringElement.anchor - position of the anchor within the element; default value (-1, -1) means that the anchor is at the element center.iterations - number of times erosion is applied.borderType - pixel extrapolation method, see #BorderTypes. #BORDER\_WRAP is not supported.borderValue - border value in case of a constant border SEE: dilate, morphologyEx, getStructuringElement

#### fillConvexPoly public static void fillConvexPoly([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) points, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Fills a convex polygon. The function cv::fillConvexPoly draws a filled convex polygon. This function is much faster than the function #fillPoly . It can fill not only convex polygons but any monotonic polygon without self-intersections, that is, a polygon whose contour intersects every horizontal line (scan line) twice at the most (though, its top-most and/or the bottom edge could be horizontal).Parameters:img - Image.points - Polygon vertices.color - Polygon color.

#### fillConvexPoly public static void fillConvexPoly([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) points, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int lineType) Fills a convex polygon. The function cv::fillConvexPoly draws a filled convex polygon. This function is much faster than the function #fillPoly . It can fill not only convex polygons but any monotonic polygon without self-intersections, that is, a polygon whose contour intersects every horizontal line (scan line) twice at the most (though, its top-most and/or the bottom edge could be horizontal).Parameters:img - Image.points - Polygon vertices.color - Polygon color.lineType - Type of the polygon boundaries. See #LineTypes

#### fillConvexPoly public static void fillConvexPoly([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) points, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int lineType, int shift) Fills a convex polygon. The function cv::fillConvexPoly draws a filled convex polygon. This function is much faster than the function #fillPoly . It can fill not only convex polygons but any monotonic polygon without self-intersections, that is, a polygon whose contour intersects every horizontal line (scan line) twice at the most (though, its top-most and/or the bottom edge could be horizontal).Parameters:img - Image.points - Polygon vertices.color - Polygon color.lineType - Type of the polygon boundaries. See #LineTypesshift - Number of fractional bits in the vertex coordinates.

#### fillPoly public static void fillPoly([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> pts, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Fills the area bounded by one or more polygons. The function cv::fillPoly fills an area bounded by several polygonal contours. The function can fill complex areas, for example, areas with holes, contours with self-intersections (some of their parts), and so forth.Parameters:img - Image.pts - Array of polygons where each polygon is represented as an array of points.color - Polygon color.

#### fillPoly public static void fillPoly([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> pts, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int lineType) Fills the area bounded by one or more polygons. The function cv::fillPoly fills an area bounded by several polygonal contours. The function can fill complex areas, for example, areas with holes, contours with self-intersections (some of their parts), and so forth.Parameters:img - Image.pts - Array of polygons where each polygon is represented as an array of points.color - Polygon color.lineType - Type of the polygon boundaries. See #LineTypes

#### fillPoly public static void fillPoly([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> pts, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int lineType, int shift) Fills the area bounded by one or more polygons. The function cv::fillPoly fills an area bounded by several polygonal contours. The function can fill complex areas, for example, areas with holes, contours with self-intersections (some of their parts), and so forth.Parameters:img - Image.pts - Array of polygons where each polygon is represented as an array of points.color - Polygon color.lineType - Type of the polygon boundaries. See #LineTypesshift - Number of fractional bits in the vertex coordinates.

#### fillPoly public static void fillPoly([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> pts, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int lineType, int shift, [Point](http://docs.google.com/org/opencv/core/Point.html) offset) Fills the area bounded by one or more polygons. The function cv::fillPoly fills an area bounded by several polygonal contours. The function can fill complex areas, for example, areas with holes, contours with self-intersections (some of their parts), and so forth.Parameters:img - Image.pts - Array of polygons where each polygon is represented as an array of points.color - Polygon color.lineType - Type of the polygon boundaries. See #LineTypesshift - Number of fractional bits in the vertex coordinates.offset - Optional offset of all points of the contours.

#### filter2D public static void filter2D([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel) Convolves an image with the kernel. The function applies an arbitrary linear filter to an image. In-place operation is supported. When the aperture is partially outside the image, the function interpolates outlier pixel values according to the specified border mode. The function does actually compute correlation, not the convolution: \(\texttt{dst} (x,y) = \sum \_{ \substack{0\leq x' < \texttt{kernel.cols}\\{0\leq y' < \texttt{kernel.rows}}}} \texttt{kernel} (x',y')\* \texttt{src} (x+x'- \texttt{anchor.x} ,y+y'- \texttt{anchor.y} )\) That is, the kernel is not mirrored around the anchor point. If you need a real convolution, flip the kernel using #flip and set the new anchor to `(kernel.cols - anchor.x - 1, kernel.rows - anchor.y - 1)`. The function uses the DFT-based algorithm in case of sufficiently large kernels (~11 x 11 or larger) and the direct algorithm for small kernels.Parameters:src - input image.dst - output image of the same size and the same number of channels as src.ddepth - desired depth of the destination image, see REF: filter\_depths "combinations"kernel - convolution kernel (or rather a correlation kernel), a single-channel floating point matrix; if you want to apply different kernels to different channels, split the image into separate color planes using split and process them individually. the kernel; the anchor should lie within the kernel; default value (-1,-1) means that the anchor is at the kernel center. SEE: sepFilter2D, dft, matchTemplate

#### filter2D public static void filter2D([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor) Convolves an image with the kernel. The function applies an arbitrary linear filter to an image. In-place operation is supported. When the aperture is partially outside the image, the function interpolates outlier pixel values according to the specified border mode. The function does actually compute correlation, not the convolution: \(\texttt{dst} (x,y) = \sum \_{ \substack{0\leq x' < \texttt{kernel.cols}\\{0\leq y' < \texttt{kernel.rows}}}} \texttt{kernel} (x',y')\* \texttt{src} (x+x'- \texttt{anchor.x} ,y+y'- \texttt{anchor.y} )\) That is, the kernel is not mirrored around the anchor point. If you need a real convolution, flip the kernel using #flip and set the new anchor to `(kernel.cols - anchor.x - 1, kernel.rows - anchor.y - 1)`. The function uses the DFT-based algorithm in case of sufficiently large kernels (~11 x 11 or larger) and the direct algorithm for small kernels.Parameters:src - input image.dst - output image of the same size and the same number of channels as src.ddepth - desired depth of the destination image, see REF: filter\_depths "combinations"kernel - convolution kernel (or rather a correlation kernel), a single-channel floating point matrix; if you want to apply different kernels to different channels, split the image into separate color planes using split and process them individually.anchor - anchor of the kernel that indicates the relative position of a filtered point within the kernel; the anchor should lie within the kernel; default value (-1,-1) means that the anchor is at the kernel center. SEE: sepFilter2D, dft, matchTemplate

#### filter2D public static void filter2D([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, double delta) Convolves an image with the kernel. The function applies an arbitrary linear filter to an image. In-place operation is supported. When the aperture is partially outside the image, the function interpolates outlier pixel values according to the specified border mode. The function does actually compute correlation, not the convolution: \(\texttt{dst} (x,y) = \sum \_{ \substack{0\leq x' < \texttt{kernel.cols}\\{0\leq y' < \texttt{kernel.rows}}}} \texttt{kernel} (x',y')\* \texttt{src} (x+x'- \texttt{anchor.x} ,y+y'- \texttt{anchor.y} )\) That is, the kernel is not mirrored around the anchor point. If you need a real convolution, flip the kernel using #flip and set the new anchor to `(kernel.cols - anchor.x - 1, kernel.rows - anchor.y - 1)`. The function uses the DFT-based algorithm in case of sufficiently large kernels (~11 x 11 or larger) and the direct algorithm for small kernels.Parameters:src - input image.dst - output image of the same size and the same number of channels as src.ddepth - desired depth of the destination image, see REF: filter\_depths "combinations"kernel - convolution kernel (or rather a correlation kernel), a single-channel floating point matrix; if you want to apply different kernels to different channels, split the image into separate color planes using split and process them individually.anchor - anchor of the kernel that indicates the relative position of a filtered point within the kernel; the anchor should lie within the kernel; default value (-1,-1) means that the anchor is at the kernel center.delta - optional value added to the filtered pixels before storing them in dst. SEE: sepFilter2D, dft, matchTemplate

#### filter2D public static void filter2D([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, double delta, int borderType) Convolves an image with the kernel. The function applies an arbitrary linear filter to an image. In-place operation is supported. When the aperture is partially outside the image, the function interpolates outlier pixel values according to the specified border mode. The function does actually compute correlation, not the convolution: \(\texttt{dst} (x,y) = \sum \_{ \substack{0\leq x' < \texttt{kernel.cols}\\{0\leq y' < \texttt{kernel.rows}}}} \texttt{kernel} (x',y')\* \texttt{src} (x+x'- \texttt{anchor.x} ,y+y'- \texttt{anchor.y} )\) That is, the kernel is not mirrored around the anchor point. If you need a real convolution, flip the kernel using #flip and set the new anchor to `(kernel.cols - anchor.x - 1, kernel.rows - anchor.y - 1)`. The function uses the DFT-based algorithm in case of sufficiently large kernels (~11 x 11 or larger) and the direct algorithm for small kernels.Parameters:src - input image.dst - output image of the same size and the same number of channels as src.ddepth - desired depth of the destination image, see REF: filter\_depths "combinations"kernel - convolution kernel (or rather a correlation kernel), a single-channel floating point matrix; if you want to apply different kernels to different channels, split the image into separate color planes using split and process them individually.anchor - anchor of the kernel that indicates the relative position of a filtered point within the kernel; the anchor should lie within the kernel; default value (-1,-1) means that the anchor is at the kernel center.delta - optional value added to the filtered pixels before storing them in dst.borderType - pixel extrapolation method, see #BorderTypes. #BORDER\_WRAP is not supported. SEE: sepFilter2D, dft, matchTemplate

#### findContours public static void findContours([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> contours, [Mat](http://docs.google.com/org/opencv/core/Mat.html) hierarchy, int mode, int method) Finds contours in a binary image. The function retrieves contours from the binary image using the algorithm CITE: Suzuki85 . The contours are a useful tool for shape analysis and object detection and recognition. See squares.cpp in the OpenCV sample directory. **Note:** Since opencv 3.2 source image is not modified by this function.Parameters:image - Source, an 8-bit single-channel image. Non-zero pixels are treated as 1's. Zero pixels remain 0's, so the image is treated as binary . You can use #compare, #inRange, #threshold , #adaptiveThreshold, #Canny, and others to create a binary image out of a grayscale or color one. If mode equals to #RETR\_CCOMP or #RETR\_FLOODFILL, the input can also be a 32-bit integer image of labels (CV\_32SC1).contours - Detected contours. Each contour is stored as a vector of points (e.g. std::vector<std::vector<cv::Point> >).hierarchy - Optional output vector (e.g. std::vector<cv::Vec4i>), containing information about the image topology. It has as many elements as the number of contours. For each i-th contour contours[i], the elements hierarchy[i][0] , hierarchy[i][1] , hierarchy[i][2] , and hierarchy[i][3] are set to 0-based indices in contours of the next and previous contours at the same hierarchical level, the first child contour and the parent contour, respectively. If for the contour i there are no next, previous, parent, or nested contours, the corresponding elements of hierarchy[i] will be negative.mode - Contour retrieval mode, see #RetrievalModesmethod - Contour approximation method, see #ContourApproximationModes contours are extracted from the image ROI and then they should be analyzed in the whole image context.

#### findContours public static void findContours([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> contours, [Mat](http://docs.google.com/org/opencv/core/Mat.html) hierarchy, int mode, int method, [Point](http://docs.google.com/org/opencv/core/Point.html) offset) Finds contours in a binary image. The function retrieves contours from the binary image using the algorithm CITE: Suzuki85 . The contours are a useful tool for shape analysis and object detection and recognition. See squares.cpp in the OpenCV sample directory. **Note:** Since opencv 3.2 source image is not modified by this function.Parameters:image - Source, an 8-bit single-channel image. Non-zero pixels are treated as 1's. Zero pixels remain 0's, so the image is treated as binary . You can use #compare, #inRange, #threshold , #adaptiveThreshold, #Canny, and others to create a binary image out of a grayscale or color one. If mode equals to #RETR\_CCOMP or #RETR\_FLOODFILL, the input can also be a 32-bit integer image of labels (CV\_32SC1).contours - Detected contours. Each contour is stored as a vector of points (e.g. std::vector<std::vector<cv::Point> >).hierarchy - Optional output vector (e.g. std::vector<cv::Vec4i>), containing information about the image topology. It has as many elements as the number of contours. For each i-th contour contours[i], the elements hierarchy[i][0] , hierarchy[i][1] , hierarchy[i][2] , and hierarchy[i][3] are set to 0-based indices in contours of the next and previous contours at the same hierarchical level, the first child contour and the parent contour, respectively. If for the contour i there are no next, previous, parent, or nested contours, the corresponding elements of hierarchy[i] will be negative.mode - Contour retrieval mode, see #RetrievalModesmethod - Contour approximation method, see #ContourApproximationModesoffset - Optional offset by which every contour point is shifted. This is useful if the contours are extracted from the image ROI and then they should be analyzed in the whole image context.

#### fitEllipse public static [RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) fitEllipse([MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) points) Fits an ellipse around a set of 2D points. The function calculates the ellipse that fits (in a least-squares sense) a set of 2D points best of all. It returns the rotated rectangle in which the ellipse is inscribed. The first algorithm described by CITE: Fitzgibbon95 is used. Developer should keep in mind that it is possible that the returned ellipse/rotatedRect data contains negative indices, due to the data points being close to the border of the containing Mat element.Parameters:points - Input 2D point set, stored in std::vector<> or Mat Returns:automatically generated

#### fitEllipseAMS public static [RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) fitEllipseAMS([Mat](http://docs.google.com/org/opencv/core/Mat.html) points) Fits an ellipse around a set of 2D points. The function calculates the ellipse that fits a set of 2D points. It returns the rotated rectangle in which the ellipse is inscribed. The Approximate Mean Square (AMS) proposed by CITE: Taubin1991 is used. For an ellipse, this basis set is \( \chi= \left(x^2, x y, y^2, x, y, 1\right) \), which is a set of six free coefficients \( A^T=\left\{A\_{\text{xx}},A\_{\text{xy}},A\_{\text{yy}},A\_x,A\_y,A\_0\right\} \). However, to specify an ellipse, all that is needed is five numbers; the major and minor axes lengths \( (a,b) \), the position \( (x\_0,y\_0) \), and the orientation \( \theta \). This is because the basis set includes lines, quadratics, parabolic and hyperbolic functions as well as elliptical functions as possible fits. If the fit is found to be a parabolic or hyperbolic function then the standard #fitEllipse method is used. The AMS method restricts the fit to parabolic, hyperbolic and elliptical curves by imposing the condition that \( A^T ( D\_x^T D\_x + D\_y^T D\_y) A = 1 \) where the matrices \( Dx \) and \( Dy \) are the partial derivatives of the design matrix \( D \) with respect to x and y. The matrices are formed row by row applying the following to each of the points in the set: \(align\*}{ D(i,:)&=\left\{x\_i^2, x\_i y\_i, y\_i^2, x\_i, y\_i, 1\right\} & D\_x(i,:)&=\left\{2 x\_i,y\_i,0,1,0,0\right\} & D\_y(i,:)&=\left\{0,x\_i,2 y\_i,0,1,0\right\} \) The AMS method minimizes the cost function \(equation\*}{ \epsilon ^2=\frac{ A^T D^T D A }{ A^T (D\_x^T D\_x + D\_y^T D\_y) A^T } \) The minimum cost is found by solving the generalized eigenvalue problem. \(equation\*}{ D^T D A = \lambda \left( D\_x^T D\_x + D\_y^T D\_y\right) A \)Parameters:points - Input 2D point set, stored in std::vector<> or Mat Returns:automatically generated

#### fitEllipseDirect public static [RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) fitEllipseDirect([Mat](http://docs.google.com/org/opencv/core/Mat.html) points) Fits an ellipse around a set of 2D points. The function calculates the ellipse that fits a set of 2D points. It returns the rotated rectangle in which the ellipse is inscribed. The Direct least square (Direct) method by CITE: Fitzgibbon1999 is used. For an ellipse, this basis set is \( \chi= \left(x^2, x y, y^2, x, y, 1\right) \), which is a set of six free coefficients \( A^T=\left\{A\_{\text{xx}},A\_{\text{xy}},A\_{\text{yy}},A\_x,A\_y,A\_0\right\} \). However, to specify an ellipse, all that is needed is five numbers; the major and minor axes lengths \( (a,b) \), the position \( (x\_0,y\_0) \), and the orientation \( \theta \). This is because the basis set includes lines, quadratics, parabolic and hyperbolic functions as well as elliptical functions as possible fits. The Direct method confines the fit to ellipses by ensuring that \( 4 A\_{xx} A\_{yy}- A\_{xy}^2 > 0 \). The condition imposed is that \( 4 A\_{xx} A\_{yy}- A\_{xy}^2=1 \) which satisfies the inequality and as the coefficients can be arbitrarily scaled is not overly restrictive. \(equation\*}{ \epsilon ^2= A^T D^T D A \quad \text{with} \quad A^T C A =1 \quad \text{and} \quad C=\left(\begin{matrix} 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \right) \) The minimum cost is found by solving the generalized eigenvalue problem. \(equation\*}{ D^T D A = \lambda \left( C\right) A \) The system produces only one positive eigenvalue \( \lambda\) which is chosen as the solution with its eigenvector \(\mathbf{u}\). These are used to find the coefficients \(equation\*}{ A = \sqrt{\frac{1}{\mathbf{u}^T C \mathbf{u}}} \mathbf{u} \) The scaling factor guarantees that \(A^T C A =1\).Parameters:points - Input 2D point set, stored in std::vector<> or Mat Returns:automatically generated

#### fitLine public static void fitLine([Mat](http://docs.google.com/org/opencv/core/Mat.html) points, [Mat](http://docs.google.com/org/opencv/core/Mat.html) line, int distType, double param, double reps, double aeps)

Fits a line to a 2D or 3D point set. The function fitLine fits a line to a 2D or 3D point set by minimizing \(\sum\_i \rho(r\_i)\) where \(r\_i\) is a distance between the \(i^{th}\) point, the line and \(\rho(r)\) is a distance function, one of the following:

* + - * DIST\_L2 \(\rho (r) = r^2/2 \quad \text{(the simplest and the fastest least-squares method)}\)
      * DIST\_L1 \(\rho (r) = r\)
      * DIST\_L12 \(\rho (r) = 2 \cdot ( \sqrt{1 + \frac{r^2}{2}} - 1)\)
      * DIST\_FAIR \(\rho \left (r \right ) = C^2 \cdot \left ( \frac{r}{C} - \log{\left(1 + \frac{r}{C}\right)} \right ) \quad \text{where} \quad C=1.3998\)
      * DIST\_WELSCH \(\rho \left (r \right ) = \frac{C^2}{2} \cdot \left ( 1 - \exp{\left(-\left(\frac{r}{C}\right)^2\right)} \right ) \quad \text{where} \quad C=2.9846\)
      * DIST\_HUBER \(\rho (r) = \fork{r^2/2}{if \(r < C\)}{C \cdot (r-C/2)}{otherwise} \quad \text{where} \quad C=1.345\)

The algorithm is based on the M-estimator ( <http://en.wikipedia.org/wiki/M-estimator> ) technique that iteratively fits the line using the weighted least-squares algorithm. After each iteration the weights \(w\_i\) are adjusted to be inversely proportional to \(\rho(r\_i)\) .Parameters:points - Input vector of 2D or 3D points, stored in std::vector<> or Mat.line - Output line parameters. In case of 2D fitting, it should be a vector of 4 elements (like Vec4f) - (vx, vy, x0, y0), where (vx, vy) is a normalized vector collinear to the line and (x0, y0) is a point on the line. In case of 3D fitting, it should be a vector of 6 elements (like Vec6f) - (vx, vy, vz, x0, y0, z0), where (vx, vy, vz) is a normalized vector collinear to the line and (x0, y0, z0) is a point on the line.distType - Distance used by the M-estimator, see #DistanceTypesparam - Numerical parameter ( C ) for some types of distances. If it is 0, an optimal value is chosen.reps - Sufficient accuracy for the radius (distance between the coordinate origin and the line).aeps - Sufficient accuracy for the angle. 0.01 would be a good default value for reps and aeps.

#### floodFill public static int floodFill([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Point](http://docs.google.com/org/opencv/core/Point.html) seedPoint, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) newVal)

Fills a connected component with the given color. The function cv::floodFill fills a connected component starting from the seed point with the specified color. The connectivity is determined by the color/brightness closeness of the neighbor pixels. The pixel at \((x,y)\) is considered to belong to the repainted domain if:

* + - * in case of a grayscale image and floating range \(\texttt{src} (x',y')- \texttt{loDiff} \leq \texttt{src} (x,y) \leq \texttt{src} (x',y')+ \texttt{upDiff}\)
      * in case of a grayscale image and fixed range \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)- \texttt{loDiff} \leq \texttt{src} (x,y) \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)+ \texttt{upDiff}\)
      * in case of a color image and floating range \(\texttt{src} (x',y')\_r- \texttt{loDiff} \_r \leq \texttt{src} (x,y)\_r \leq \texttt{src} (x',y')\_r+ \texttt{upDiff} \_r,\) \(\texttt{src} (x',y')\_g- \texttt{loDiff} \_g \leq \texttt{src} (x,y)\_g \leq \texttt{src} (x',y')\_g+ \texttt{upDiff} \_g\) and \(\texttt{src} (x',y')\_b- \texttt{loDiff} \_b \leq \texttt{src} (x,y)\_b \leq \texttt{src} (x',y')\_b+ \texttt{upDiff} \_b\)
      * in case of a color image and fixed range \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_r- \texttt{loDiff} \_r \leq \texttt{src} (x,y)\_r \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_r+ \texttt{upDiff} \_r,\) \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_g- \texttt{loDiff} \_g \leq \texttt{src} (x,y)\_g \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_g+ \texttt{upDiff} \_g\) and \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_b- \texttt{loDiff} \_b \leq \texttt{src} (x,y)\_b \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_b+ \texttt{upDiff} \_b\)

where \(src(x',y')\) is the value of one of pixel neighbors that is already known to belong to the component. That is, to be added to the connected component, a color/brightness of the pixel should be close enough to:

* + - * Color/brightness of one of its neighbors that already belong to the connected component in case of a floating range.
      * Color/brightness of the seed point in case of a fixed range.

Use these functions to either mark a connected component with the specified color in-place, or build a mask and then extract the contour, or copy the region to another image, and so on.Parameters:image - Input/output 1- or 3-channel, 8-bit, or floating-point image. It is modified by the function unless the #FLOODFILL\_MASK\_ONLY flag is set in the second variant of the function. See the details below.mask - Operation mask that should be a single-channel 8-bit image, 2 pixels wider and 2 pixels taller than image. Since this is both an input and output parameter, you must take responsibility of initializing it. Flood-filling cannot go across non-zero pixels in the input mask. For example, an edge detector output can be used as a mask to stop filling at edges. On output, pixels in the mask corresponding to filled pixels in the image are set to 1 or to the a value specified in flags as described below. Additionally, the function fills the border of the mask with ones to simplify internal processing. It is therefore possible to use the same mask in multiple calls to the function to make sure the filled areas do not overlap.seedPoint - Starting point.newVal - New value of the repainted domain pixels. one of its neighbors belonging to the component, or a seed pixel being added to the component. one of its neighbors belonging to the component, or a seed pixel being added to the component. repainted domain. 4 means that only the four nearest neighbor pixels (those that share an edge) are considered. A connectivity value of 8 means that the eight nearest neighbor pixels (those that share a corner) will be considered. The next 8 bits (8-16) contain a value between 1 and 255 with which to fill the mask (the default value is 1). For example, 4 | ( 255 << 8 ) will consider 4 nearest neighbours and fill the mask with a value of 255. The following additional options occupy higher bits and therefore may be further combined with the connectivity and mask fill values using bit-wise or (|), see #FloodFillFlags. **Note:** Since the mask is larger than the filled image, a pixel \((x, y)\) in image corresponds to the pixel \((x+1, y+1)\) in the mask . SEE: findContours Returns:automatically generated

#### floodFill public static int floodFill([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Point](http://docs.google.com/org/opencv/core/Point.html) seedPoint, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) newVal, [Rect](http://docs.google.com/org/opencv/core/Rect.html) rect)

Fills a connected component with the given color. The function cv::floodFill fills a connected component starting from the seed point with the specified color. The connectivity is determined by the color/brightness closeness of the neighbor pixels. The pixel at \((x,y)\) is considered to belong to the repainted domain if:

* + - * in case of a grayscale image and floating range \(\texttt{src} (x',y')- \texttt{loDiff} \leq \texttt{src} (x,y) \leq \texttt{src} (x',y')+ \texttt{upDiff}\)
      * in case of a grayscale image and fixed range \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)- \texttt{loDiff} \leq \texttt{src} (x,y) \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)+ \texttt{upDiff}\)
      * in case of a color image and floating range \(\texttt{src} (x',y')\_r- \texttt{loDiff} \_r \leq \texttt{src} (x,y)\_r \leq \texttt{src} (x',y')\_r+ \texttt{upDiff} \_r,\) \(\texttt{src} (x',y')\_g- \texttt{loDiff} \_g \leq \texttt{src} (x,y)\_g \leq \texttt{src} (x',y')\_g+ \texttt{upDiff} \_g\) and \(\texttt{src} (x',y')\_b- \texttt{loDiff} \_b \leq \texttt{src} (x,y)\_b \leq \texttt{src} (x',y')\_b+ \texttt{upDiff} \_b\)
      * in case of a color image and fixed range \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_r- \texttt{loDiff} \_r \leq \texttt{src} (x,y)\_r \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_r+ \texttt{upDiff} \_r,\) \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_g- \texttt{loDiff} \_g \leq \texttt{src} (x,y)\_g \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_g+ \texttt{upDiff} \_g\) and \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_b- \texttt{loDiff} \_b \leq \texttt{src} (x,y)\_b \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_b+ \texttt{upDiff} \_b\)

where \(src(x',y')\) is the value of one of pixel neighbors that is already known to belong to the component. That is, to be added to the connected component, a color/brightness of the pixel should be close enough to:

* + - * Color/brightness of one of its neighbors that already belong to the connected component in case of a floating range.
      * Color/brightness of the seed point in case of a fixed range.

Use these functions to either mark a connected component with the specified color in-place, or build a mask and then extract the contour, or copy the region to another image, and so on.Parameters:image - Input/output 1- or 3-channel, 8-bit, or floating-point image. It is modified by the function unless the #FLOODFILL\_MASK\_ONLY flag is set in the second variant of the function. See the details below.mask - Operation mask that should be a single-channel 8-bit image, 2 pixels wider and 2 pixels taller than image. Since this is both an input and output parameter, you must take responsibility of initializing it. Flood-filling cannot go across non-zero pixels in the input mask. For example, an edge detector output can be used as a mask to stop filling at edges. On output, pixels in the mask corresponding to filled pixels in the image are set to 1 or to the a value specified in flags as described below. Additionally, the function fills the border of the mask with ones to simplify internal processing. It is therefore possible to use the same mask in multiple calls to the function to make sure the filled areas do not overlap.seedPoint - Starting point.newVal - New value of the repainted domain pixels. one of its neighbors belonging to the component, or a seed pixel being added to the component. one of its neighbors belonging to the component, or a seed pixel being added to the component.rect - Optional output parameter set by the function to the minimum bounding rectangle of the repainted domain. 4 means that only the four nearest neighbor pixels (those that share an edge) are considered. A connectivity value of 8 means that the eight nearest neighbor pixels (those that share a corner) will be considered. The next 8 bits (8-16) contain a value between 1 and 255 with which to fill the mask (the default value is 1). For example, 4 | ( 255 << 8 ) will consider 4 nearest neighbours and fill the mask with a value of 255. The following additional options occupy higher bits and therefore may be further combined with the connectivity and mask fill values using bit-wise or (|), see #FloodFillFlags. **Note:** Since the mask is larger than the filled image, a pixel \((x, y)\) in image corresponds to the pixel \((x+1, y+1)\) in the mask . SEE: findContours Returns:automatically generated

#### floodFill public static int floodFill([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Point](http://docs.google.com/org/opencv/core/Point.html) seedPoint, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) newVal, [Rect](http://docs.google.com/org/opencv/core/Rect.html) rect, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) loDiff)

Fills a connected component with the given color. The function cv::floodFill fills a connected component starting from the seed point with the specified color. The connectivity is determined by the color/brightness closeness of the neighbor pixels. The pixel at \((x,y)\) is considered to belong to the repainted domain if:

* + - * in case of a grayscale image and floating range \(\texttt{src} (x',y')- \texttt{loDiff} \leq \texttt{src} (x,y) \leq \texttt{src} (x',y')+ \texttt{upDiff}\)
      * in case of a grayscale image and fixed range \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)- \texttt{loDiff} \leq \texttt{src} (x,y) \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)+ \texttt{upDiff}\)
      * in case of a color image and floating range \(\texttt{src} (x',y')\_r- \texttt{loDiff} \_r \leq \texttt{src} (x,y)\_r \leq \texttt{src} (x',y')\_r+ \texttt{upDiff} \_r,\) \(\texttt{src} (x',y')\_g- \texttt{loDiff} \_g \leq \texttt{src} (x,y)\_g \leq \texttt{src} (x',y')\_g+ \texttt{upDiff} \_g\) and \(\texttt{src} (x',y')\_b- \texttt{loDiff} \_b \leq \texttt{src} (x,y)\_b \leq \texttt{src} (x',y')\_b+ \texttt{upDiff} \_b\)
      * in case of a color image and fixed range \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_r- \texttt{loDiff} \_r \leq \texttt{src} (x,y)\_r \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_r+ \texttt{upDiff} \_r,\) \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_g- \texttt{loDiff} \_g \leq \texttt{src} (x,y)\_g \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_g+ \texttt{upDiff} \_g\) and \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_b- \texttt{loDiff} \_b \leq \texttt{src} (x,y)\_b \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_b+ \texttt{upDiff} \_b\)

where \(src(x',y')\) is the value of one of pixel neighbors that is already known to belong to the component. That is, to be added to the connected component, a color/brightness of the pixel should be close enough to:

* + - * Color/brightness of one of its neighbors that already belong to the connected component in case of a floating range.
      * Color/brightness of the seed point in case of a fixed range.

Use these functions to either mark a connected component with the specified color in-place, or build a mask and then extract the contour, or copy the region to another image, and so on.Parameters:image - Input/output 1- or 3-channel, 8-bit, or floating-point image. It is modified by the function unless the #FLOODFILL\_MASK\_ONLY flag is set in the second variant of the function. See the details below.mask - Operation mask that should be a single-channel 8-bit image, 2 pixels wider and 2 pixels taller than image. Since this is both an input and output parameter, you must take responsibility of initializing it. Flood-filling cannot go across non-zero pixels in the input mask. For example, an edge detector output can be used as a mask to stop filling at edges. On output, pixels in the mask corresponding to filled pixels in the image are set to 1 or to the a value specified in flags as described below. Additionally, the function fills the border of the mask with ones to simplify internal processing. It is therefore possible to use the same mask in multiple calls to the function to make sure the filled areas do not overlap.seedPoint - Starting point.newVal - New value of the repainted domain pixels.loDiff - Maximal lower brightness/color difference between the currently observed pixel and one of its neighbors belonging to the component, or a seed pixel being added to the component. one of its neighbors belonging to the component, or a seed pixel being added to the component.rect - Optional output parameter set by the function to the minimum bounding rectangle of the repainted domain. 4 means that only the four nearest neighbor pixels (those that share an edge) are considered. A connectivity value of 8 means that the eight nearest neighbor pixels (those that share a corner) will be considered. The next 8 bits (8-16) contain a value between 1 and 255 with which to fill the mask (the default value is 1). For example, 4 | ( 255 << 8 ) will consider 4 nearest neighbours and fill the mask with a value of 255. The following additional options occupy higher bits and therefore may be further combined with the connectivity and mask fill values using bit-wise or (|), see #FloodFillFlags. **Note:** Since the mask is larger than the filled image, a pixel \((x, y)\) in image corresponds to the pixel \((x+1, y+1)\) in the mask . SEE: findContours Returns:automatically generated

#### floodFill public static int floodFill([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Point](http://docs.google.com/org/opencv/core/Point.html) seedPoint, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) newVal, [Rect](http://docs.google.com/org/opencv/core/Rect.html) rect, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) loDiff, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) upDiff)

Fills a connected component with the given color. The function cv::floodFill fills a connected component starting from the seed point with the specified color. The connectivity is determined by the color/brightness closeness of the neighbor pixels. The pixel at \((x,y)\) is considered to belong to the repainted domain if:

* + - * in case of a grayscale image and floating range \(\texttt{src} (x',y')- \texttt{loDiff} \leq \texttt{src} (x,y) \leq \texttt{src} (x',y')+ \texttt{upDiff}\)
      * in case of a grayscale image and fixed range \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)- \texttt{loDiff} \leq \texttt{src} (x,y) \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)+ \texttt{upDiff}\)
      * in case of a color image and floating range \(\texttt{src} (x',y')\_r- \texttt{loDiff} \_r \leq \texttt{src} (x,y)\_r \leq \texttt{src} (x',y')\_r+ \texttt{upDiff} \_r,\) \(\texttt{src} (x',y')\_g- \texttt{loDiff} \_g \leq \texttt{src} (x,y)\_g \leq \texttt{src} (x',y')\_g+ \texttt{upDiff} \_g\) and \(\texttt{src} (x',y')\_b- \texttt{loDiff} \_b \leq \texttt{src} (x,y)\_b \leq \texttt{src} (x',y')\_b+ \texttt{upDiff} \_b\)
      * in case of a color image and fixed range \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_r- \texttt{loDiff} \_r \leq \texttt{src} (x,y)\_r \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_r+ \texttt{upDiff} \_r,\) \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_g- \texttt{loDiff} \_g \leq \texttt{src} (x,y)\_g \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_g+ \texttt{upDiff} \_g\) and \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_b- \texttt{loDiff} \_b \leq \texttt{src} (x,y)\_b \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_b+ \texttt{upDiff} \_b\)

where \(src(x',y')\) is the value of one of pixel neighbors that is already known to belong to the component. That is, to be added to the connected component, a color/brightness of the pixel should be close enough to:

* + - * Color/brightness of one of its neighbors that already belong to the connected component in case of a floating range.
      * Color/brightness of the seed point in case of a fixed range.

Use these functions to either mark a connected component with the specified color in-place, or build a mask and then extract the contour, or copy the region to another image, and so on.Parameters:image - Input/output 1- or 3-channel, 8-bit, or floating-point image. It is modified by the function unless the #FLOODFILL\_MASK\_ONLY flag is set in the second variant of the function. See the details below.mask - Operation mask that should be a single-channel 8-bit image, 2 pixels wider and 2 pixels taller than image. Since this is both an input and output parameter, you must take responsibility of initializing it. Flood-filling cannot go across non-zero pixels in the input mask. For example, an edge detector output can be used as a mask to stop filling at edges. On output, pixels in the mask corresponding to filled pixels in the image are set to 1 or to the a value specified in flags as described below. Additionally, the function fills the border of the mask with ones to simplify internal processing. It is therefore possible to use the same mask in multiple calls to the function to make sure the filled areas do not overlap.seedPoint - Starting point.newVal - New value of the repainted domain pixels.loDiff - Maximal lower brightness/color difference between the currently observed pixel and one of its neighbors belonging to the component, or a seed pixel being added to the component.upDiff - Maximal upper brightness/color difference between the currently observed pixel and one of its neighbors belonging to the component, or a seed pixel being added to the component.rect - Optional output parameter set by the function to the minimum bounding rectangle of the repainted domain. 4 means that only the four nearest neighbor pixels (those that share an edge) are considered. A connectivity value of 8 means that the eight nearest neighbor pixels (those that share a corner) will be considered. The next 8 bits (8-16) contain a value between 1 and 255 with which to fill the mask (the default value is 1). For example, 4 | ( 255 << 8 ) will consider 4 nearest neighbours and fill the mask with a value of 255. The following additional options occupy higher bits and therefore may be further combined with the connectivity and mask fill values using bit-wise or (|), see #FloodFillFlags. **Note:** Since the mask is larger than the filled image, a pixel \((x, y)\) in image corresponds to the pixel \((x+1, y+1)\) in the mask . SEE: findContours Returns:automatically generated

#### floodFill public static int floodFill([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Point](http://docs.google.com/org/opencv/core/Point.html) seedPoint, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) newVal, [Rect](http://docs.google.com/org/opencv/core/Rect.html) rect, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) loDiff, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) upDiff, int flags)

Fills a connected component with the given color. The function cv::floodFill fills a connected component starting from the seed point with the specified color. The connectivity is determined by the color/brightness closeness of the neighbor pixels. The pixel at \((x,y)\) is considered to belong to the repainted domain if:

* + - * in case of a grayscale image and floating range \(\texttt{src} (x',y')- \texttt{loDiff} \leq \texttt{src} (x,y) \leq \texttt{src} (x',y')+ \texttt{upDiff}\)
      * in case of a grayscale image and fixed range \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)- \texttt{loDiff} \leq \texttt{src} (x,y) \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)+ \texttt{upDiff}\)
      * in case of a color image and floating range \(\texttt{src} (x',y')\_r- \texttt{loDiff} \_r \leq \texttt{src} (x,y)\_r \leq \texttt{src} (x',y')\_r+ \texttt{upDiff} \_r,\) \(\texttt{src} (x',y')\_g- \texttt{loDiff} \_g \leq \texttt{src} (x,y)\_g \leq \texttt{src} (x',y')\_g+ \texttt{upDiff} \_g\) and \(\texttt{src} (x',y')\_b- \texttt{loDiff} \_b \leq \texttt{src} (x,y)\_b \leq \texttt{src} (x',y')\_b+ \texttt{upDiff} \_b\)
      * in case of a color image and fixed range \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_r- \texttt{loDiff} \_r \leq \texttt{src} (x,y)\_r \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_r+ \texttt{upDiff} \_r,\) \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_g- \texttt{loDiff} \_g \leq \texttt{src} (x,y)\_g \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_g+ \texttt{upDiff} \_g\) and \(\texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_b- \texttt{loDiff} \_b \leq \texttt{src} (x,y)\_b \leq \texttt{src} ( \texttt{seedPoint} .x, \texttt{seedPoint} .y)\_b+ \texttt{upDiff} \_b\)

where \(src(x',y')\) is the value of one of pixel neighbors that is already known to belong to the component. That is, to be added to the connected component, a color/brightness of the pixel should be close enough to:

* + - * Color/brightness of one of its neighbors that already belong to the connected component in case of a floating range.
      * Color/brightness of the seed point in case of a fixed range.

Use these functions to either mark a connected component with the specified color in-place, or build a mask and then extract the contour, or copy the region to another image, and so on.Parameters:image - Input/output 1- or 3-channel, 8-bit, or floating-point image. It is modified by the function unless the #FLOODFILL\_MASK\_ONLY flag is set in the second variant of the function. See the details below.mask - Operation mask that should be a single-channel 8-bit image, 2 pixels wider and 2 pixels taller than image. Since this is both an input and output parameter, you must take responsibility of initializing it. Flood-filling cannot go across non-zero pixels in the input mask. For example, an edge detector output can be used as a mask to stop filling at edges. On output, pixels in the mask corresponding to filled pixels in the image are set to 1 or to the a value specified in flags as described below. Additionally, the function fills the border of the mask with ones to simplify internal processing. It is therefore possible to use the same mask in multiple calls to the function to make sure the filled areas do not overlap.seedPoint - Starting point.newVal - New value of the repainted domain pixels.loDiff - Maximal lower brightness/color difference between the currently observed pixel and one of its neighbors belonging to the component, or a seed pixel being added to the component.upDiff - Maximal upper brightness/color difference between the currently observed pixel and one of its neighbors belonging to the component, or a seed pixel being added to the component.rect - Optional output parameter set by the function to the minimum bounding rectangle of the repainted domain.flags - Operation flags. The first 8 bits contain a connectivity value. The default value of 4 means that only the four nearest neighbor pixels (those that share an edge) are considered. A connectivity value of 8 means that the eight nearest neighbor pixels (those that share a corner) will be considered. The next 8 bits (8-16) contain a value between 1 and 255 with which to fill the mask (the default value is 1). For example, 4 | ( 255 << 8 ) will consider 4 nearest neighbours and fill the mask with a value of 255. The following additional options occupy higher bits and therefore may be further combined with the connectivity and mask fill values using bit-wise or (|), see #FloodFillFlags. **Note:** Since the mask is larger than the filled image, a pixel \((x, y)\) in image corresponds to the pixel \((x+1, y+1)\) in the mask . SEE: findContours Returns:automatically generated

#### GaussianBlur public static void GaussianBlur([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, double sigmaX) Blurs an image using a Gaussian filter. The function convolves the source image with the specified Gaussian kernel. In-place filtering is supported.Parameters:src - input image; the image can have any number of channels, which are processed independently, but the depth should be CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - output image of the same size and type as src.ksize - Gaussian kernel size. ksize.width and ksize.height can differ but they both must be positive and odd. Or, they can be zero's and then they are computed from sigma.sigmaX - Gaussian kernel standard deviation in X direction. equal to sigmaX, if both sigmas are zeros, they are computed from ksize.width and ksize.height, respectively (see #getGaussianKernel for details); to fully control the result regardless of possible future modifications of all this semantics, it is recommended to specify all of ksize, sigmaX, and sigmaY. SEE: sepFilter2D, filter2D, blur, boxFilter, bilateralFilter, medianBlur

#### GaussianBlur public static void GaussianBlur([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, double sigmaX, double sigmaY) Blurs an image using a Gaussian filter. The function convolves the source image with the specified Gaussian kernel. In-place filtering is supported.Parameters:src - input image; the image can have any number of channels, which are processed independently, but the depth should be CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - output image of the same size and type as src.ksize - Gaussian kernel size. ksize.width and ksize.height can differ but they both must be positive and odd. Or, they can be zero's and then they are computed from sigma.sigmaX - Gaussian kernel standard deviation in X direction.sigmaY - Gaussian kernel standard deviation in Y direction; if sigmaY is zero, it is set to be equal to sigmaX, if both sigmas are zeros, they are computed from ksize.width and ksize.height, respectively (see #getGaussianKernel for details); to fully control the result regardless of possible future modifications of all this semantics, it is recommended to specify all of ksize, sigmaX, and sigmaY. SEE: sepFilter2D, filter2D, blur, boxFilter, bilateralFilter, medianBlur

#### GaussianBlur public static void GaussianBlur([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, double sigmaX, double sigmaY, int borderType) Blurs an image using a Gaussian filter. The function convolves the source image with the specified Gaussian kernel. In-place filtering is supported.Parameters:src - input image; the image can have any number of channels, which are processed independently, but the depth should be CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - output image of the same size and type as src.ksize - Gaussian kernel size. ksize.width and ksize.height can differ but they both must be positive and odd. Or, they can be zero's and then they are computed from sigma.sigmaX - Gaussian kernel standard deviation in X direction.sigmaY - Gaussian kernel standard deviation in Y direction; if sigmaY is zero, it is set to be equal to sigmaX, if both sigmas are zeros, they are computed from ksize.width and ksize.height, respectively (see #getGaussianKernel for details); to fully control the result regardless of possible future modifications of all this semantics, it is recommended to specify all of ksize, sigmaX, and sigmaY.borderType - pixel extrapolation method, see #BorderTypes. #BORDER\_WRAP is not supported. SEE: sepFilter2D, filter2D, blur, boxFilter, bilateralFilter, medianBlur

#### getAffineTransform public static [Mat](http://docs.google.com/org/opencv/core/Mat.html) getAffineTransform([MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) src, [MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) dst)

#### getDefaultNewCameraMatrix public static [Mat](http://docs.google.com/org/opencv/core/Mat.html) getDefaultNewCameraMatrix([Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix) Returns the default new camera matrix. The function returns the camera matrix that is either an exact copy of the input cameraMatrix (when centerPrinicipalPoint=false ), or the modified one (when centerPrincipalPoint=true). In the latter case, the new camera matrix will be: \(\begin{bmatrix} f\_x && 0 && ( \texttt{imgSize.width} -1)\*0.5 \\ 0 && f\_y && ( \texttt{imgSize.height} -1)\*0.5 \\ 0 && 0 && 1 \end{bmatrix} ,\) where \(f\_x\) and \(f\_y\) are \((0,0)\) and \((1,1)\) elements of cameraMatrix, respectively. By default, the undistortion functions in OpenCV (see #initUndistortRectifyMap, #undistort) do not move the principal point. However, when you work with stereo, it is important to move the principal points in both views to the same y-coordinate (which is required by most of stereo correspondence algorithms), and may be to the same x-coordinate too. So, you can form the new camera matrix for each view where the principal points are located at the center.Parameters:cameraMatrix - Input camera matrix. parameter indicates whether this location should be at the image center or not. Returns:automatically generated

#### getDefaultNewCameraMatrix public static [Mat](http://docs.google.com/org/opencv/core/Mat.html) getDefaultNewCameraMatrix([Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Size](http://docs.google.com/org/opencv/core/Size.html) imgsize) Returns the default new camera matrix. The function returns the camera matrix that is either an exact copy of the input cameraMatrix (when centerPrinicipalPoint=false ), or the modified one (when centerPrincipalPoint=true). In the latter case, the new camera matrix will be: \(\begin{bmatrix} f\_x && 0 && ( \texttt{imgSize.width} -1)\*0.5 \\ 0 && f\_y && ( \texttt{imgSize.height} -1)\*0.5 \\ 0 && 0 && 1 \end{bmatrix} ,\) where \(f\_x\) and \(f\_y\) are \((0,0)\) and \((1,1)\) elements of cameraMatrix, respectively. By default, the undistortion functions in OpenCV (see #initUndistortRectifyMap, #undistort) do not move the principal point. However, when you work with stereo, it is important to move the principal points in both views to the same y-coordinate (which is required by most of stereo correspondence algorithms), and may be to the same x-coordinate too. So, you can form the new camera matrix for each view where the principal points are located at the center.Parameters:cameraMatrix - Input camera matrix.imgsize - Camera view image size in pixels. parameter indicates whether this location should be at the image center or not. Returns:automatically generated

#### getDefaultNewCameraMatrix public static [Mat](http://docs.google.com/org/opencv/core/Mat.html) getDefaultNewCameraMatrix([Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Size](http://docs.google.com/org/opencv/core/Size.html) imgsize, boolean centerPrincipalPoint) Returns the default new camera matrix. The function returns the camera matrix that is either an exact copy of the input cameraMatrix (when centerPrinicipalPoint=false ), or the modified one (when centerPrincipalPoint=true). In the latter case, the new camera matrix will be: \(\begin{bmatrix} f\_x && 0 && ( \texttt{imgSize.width} -1)\*0.5 \\ 0 && f\_y && ( \texttt{imgSize.height} -1)\*0.5 \\ 0 && 0 && 1 \end{bmatrix} ,\) where \(f\_x\) and \(f\_y\) are \((0,0)\) and \((1,1)\) elements of cameraMatrix, respectively. By default, the undistortion functions in OpenCV (see #initUndistortRectifyMap, #undistort) do not move the principal point. However, when you work with stereo, it is important to move the principal points in both views to the same y-coordinate (which is required by most of stereo correspondence algorithms), and may be to the same x-coordinate too. So, you can form the new camera matrix for each view where the principal points are located at the center.Parameters:cameraMatrix - Input camera matrix.imgsize - Camera view image size in pixels.centerPrincipalPoint - Location of the principal point in the new camera matrix. The parameter indicates whether this location should be at the image center or not. Returns:automatically generated

#### getDerivKernels public static void getDerivKernels([Mat](http://docs.google.com/org/opencv/core/Mat.html) kx, [Mat](http://docs.google.com/org/opencv/core/Mat.html) ky, int dx, int dy, int ksize) Returns filter coefficients for computing spatial image derivatives. The function computes and returns the filter coefficients for spatial image derivatives. When ksize=CV\_SCHARR, the Scharr \(3 \times 3\) kernels are generated (see #Scharr). Otherwise, Sobel kernels are generated (see #Sobel). The filters are normally passed to #sepFilter2D or toParameters:kx - Output matrix of row filter coefficients. It has the type ktype .ky - Output matrix of column filter coefficients. It has the type ktype .dx - Derivative order in respect of x.dy - Derivative order in respect of y.ksize - Aperture size. It can be CV\_SCHARR, 1, 3, 5, or 7. Theoretically, the coefficients should have the denominator \(=2^{ksize\*2-dx-dy-2}\). If you are going to filter floating-point images, you are likely to use the normalized kernels. But if you compute derivatives of an 8-bit image, store the results in a 16-bit image, and wish to preserve all the fractional bits, you may want to set normalize=false .

#### getDerivKernels public static void getDerivKernels([Mat](http://docs.google.com/org/opencv/core/Mat.html) kx, [Mat](http://docs.google.com/org/opencv/core/Mat.html) ky, int dx, int dy, int ksize, boolean normalize) Returns filter coefficients for computing spatial image derivatives. The function computes and returns the filter coefficients for spatial image derivatives. When ksize=CV\_SCHARR, the Scharr \(3 \times 3\) kernels are generated (see #Scharr). Otherwise, Sobel kernels are generated (see #Sobel). The filters are normally passed to #sepFilter2D or toParameters:kx - Output matrix of row filter coefficients. It has the type ktype .ky - Output matrix of column filter coefficients. It has the type ktype .dx - Derivative order in respect of x.dy - Derivative order in respect of y.ksize - Aperture size. It can be CV\_SCHARR, 1, 3, 5, or 7.normalize - Flag indicating whether to normalize (scale down) the filter coefficients or not. Theoretically, the coefficients should have the denominator \(=2^{ksize\*2-dx-dy-2}\). If you are going to filter floating-point images, you are likely to use the normalized kernels. But if you compute derivatives of an 8-bit image, store the results in a 16-bit image, and wish to preserve all the fractional bits, you may want to set normalize=false .

#### getDerivKernels public static void getDerivKernels([Mat](http://docs.google.com/org/opencv/core/Mat.html) kx, [Mat](http://docs.google.com/org/opencv/core/Mat.html) ky, int dx, int dy, int ksize, boolean normalize, int ktype) Returns filter coefficients for computing spatial image derivatives. The function computes and returns the filter coefficients for spatial image derivatives. When ksize=CV\_SCHARR, the Scharr \(3 \times 3\) kernels are generated (see #Scharr). Otherwise, Sobel kernels are generated (see #Sobel). The filters are normally passed to #sepFilter2D or toParameters:kx - Output matrix of row filter coefficients. It has the type ktype .ky - Output matrix of column filter coefficients. It has the type ktype .dx - Derivative order in respect of x.dy - Derivative order in respect of y.ksize - Aperture size. It can be CV\_SCHARR, 1, 3, 5, or 7.normalize - Flag indicating whether to normalize (scale down) the filter coefficients or not. Theoretically, the coefficients should have the denominator \(=2^{ksize\*2-dx-dy-2}\). If you are going to filter floating-point images, you are likely to use the normalized kernels. But if you compute derivatives of an 8-bit image, store the results in a 16-bit image, and wish to preserve all the fractional bits, you may want to set normalize=false .ktype - Type of filter coefficients. It can be CV\_32f or CV\_64F .

#### getFontScaleFromHeight public static double getFontScaleFromHeight(int fontFace, int pixelHeight) Calculates the font-specific size to use to achieve a given height in pixels.Parameters:fontFace - Font to use, see cv::HersheyFonts.pixelHeight - Pixel height to compute the fontScale for Returns:The fontSize to use for cv::putText SEE: cv::putText

#### getFontScaleFromHeight public static double getFontScaleFromHeight(int fontFace, int pixelHeight, int thickness) Calculates the font-specific size to use to achieve a given height in pixels.Parameters:fontFace - Font to use, see cv::HersheyFonts.pixelHeight - Pixel height to compute the fontScale forthickness - Thickness of lines used to render the text.See putText for details. Returns:The fontSize to use for cv::putText SEE: cv::putText

#### getGaborKernel public static [Mat](http://docs.google.com/org/opencv/core/Mat.html) getGaborKernel([Size](http://docs.google.com/org/opencv/core/Size.html) ksize, double sigma, double theta, double lambd, double gamma) Returns Gabor filter coefficients. For more details about gabor filter equations and parameters, see: [Gabor Filter](http://en.wikipedia.org/wiki/Gabor\_filter).Parameters:ksize - Size of the filter returned.sigma - Standard deviation of the gaussian envelope.theta - Orientation of the normal to the parallel stripes of a Gabor function.lambd - Wavelength of the sinusoidal factor.gamma - Spatial aspect ratio. Returns:automatically generated

#### getGaborKernel public static [Mat](http://docs.google.com/org/opencv/core/Mat.html) getGaborKernel([Size](http://docs.google.com/org/opencv/core/Size.html) ksize, double sigma, double theta, double lambd, double gamma, double psi) Returns Gabor filter coefficients. For more details about gabor filter equations and parameters, see: [Gabor Filter](http://en.wikipedia.org/wiki/Gabor\_filter).Parameters:ksize - Size of the filter returned.sigma - Standard deviation of the gaussian envelope.theta - Orientation of the normal to the parallel stripes of a Gabor function.lambd - Wavelength of the sinusoidal factor.gamma - Spatial aspect ratio.psi - Phase offset. Returns:automatically generated

#### getGaborKernel public static [Mat](http://docs.google.com/org/opencv/core/Mat.html) getGaborKernel([Size](http://docs.google.com/org/opencv/core/Size.html) ksize, double sigma, double theta, double lambd, double gamma, double psi, int ktype) Returns Gabor filter coefficients. For more details about gabor filter equations and parameters, see: [Gabor Filter](http://en.wikipedia.org/wiki/Gabor\_filter).Parameters:ksize - Size of the filter returned.sigma - Standard deviation of the gaussian envelope.theta - Orientation of the normal to the parallel stripes of a Gabor function.lambd - Wavelength of the sinusoidal factor.gamma - Spatial aspect ratio.psi - Phase offset.ktype - Type of filter coefficients. It can be CV\_32F or CV\_64F . Returns:automatically generated

#### getGaussianKernel public static [Mat](http://docs.google.com/org/opencv/core/Mat.html) getGaussianKernel(int ksize, double sigma) Returns Gaussian filter coefficients. The function computes and returns the \(\texttt{ksize} \times 1\) matrix of Gaussian filter coefficients: \(G\_i= \alpha \*e^{-(i-( \texttt{ksize} -1)/2)^2/(2\* \texttt{sigma}^2)},\) where \(i=0..\texttt{ksize}-1\) and \(\alpha\) is the scale factor chosen so that \(\sum\_i G\_i=1\). Two of such generated kernels can be passed to sepFilter2D. Those functions automatically recognize smoothing kernels (a symmetrical kernel with sum of weights equal to 1) and handle them accordingly. You may also use the higher-level GaussianBlur.Parameters:ksize - Aperture size. It should be odd ( \(\texttt{ksize} \mod 2 = 1\) ) and positive.sigma - Gaussian standard deviation. If it is non-positive, it is computed from ksize as sigma = 0.3\*((ksize-1)\*0.5 - 1) + 0.8. SEE: sepFilter2D, getDerivKernels, getStructuringElement, GaussianBlur Returns:automatically generated

#### getGaussianKernel public static [Mat](http://docs.google.com/org/opencv/core/Mat.html) getGaussianKernel(int ksize, double sigma, int ktype) Returns Gaussian filter coefficients. The function computes and returns the \(\texttt{ksize} \times 1\) matrix of Gaussian filter coefficients: \(G\_i= \alpha \*e^{-(i-( \texttt{ksize} -1)/2)^2/(2\* \texttt{sigma}^2)},\) where \(i=0..\texttt{ksize}-1\) and \(\alpha\) is the scale factor chosen so that \(\sum\_i G\_i=1\). Two of such generated kernels can be passed to sepFilter2D. Those functions automatically recognize smoothing kernels (a symmetrical kernel with sum of weights equal to 1) and handle them accordingly. You may also use the higher-level GaussianBlur.Parameters:ksize - Aperture size. It should be odd ( \(\texttt{ksize} \mod 2 = 1\) ) and positive.sigma - Gaussian standard deviation. If it is non-positive, it is computed from ksize as sigma = 0.3\*((ksize-1)\*0.5 - 1) + 0.8.ktype - Type of filter coefficients. It can be CV\_32F or CV\_64F . SEE: sepFilter2D, getDerivKernels, getStructuringElement, GaussianBlur Returns:automatically generated

#### getPerspectiveTransform public static [Mat](http://docs.google.com/org/opencv/core/Mat.html) getPerspectiveTransform([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Calculates a perspective transform from four pairs of the corresponding points. The function calculates the \(3 \times 3\) matrix of a perspective transform so that: \(\begin{bmatrix} t\_i x'\_i \\ t\_i y'\_i \\ t\_i \end{bmatrix} = \texttt{map\_matrix} \cdot \begin{bmatrix} x\_i \\ y\_i \\ 1 \end{bmatrix}\) where \(dst(i)=(x'\_i,y'\_i), src(i)=(x\_i, y\_i), i=0,1,2,3\)Parameters:src - Coordinates of quadrangle vertices in the source image.dst - Coordinates of the corresponding quadrangle vertices in the destination image. SEE: findHomography, warpPerspective, perspectiveTransform Returns:automatically generated

#### getRectSubPix public static void getRectSubPix([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Size](http://docs.google.com/org/opencv/core/Size.html) patchSize, [Point](http://docs.google.com/org/opencv/core/Point.html) center, [Mat](http://docs.google.com/org/opencv/core/Mat.html) patch) Retrieves a pixel rectangle from an image with sub-pixel accuracy. The function getRectSubPix extracts pixels from src: \(patch(x, y) = src(x + \texttt{center.x} - ( \texttt{dst.cols} -1)\*0.5, y + \texttt{center.y} - ( \texttt{dst.rows} -1)\*0.5)\) where the values of the pixels at non-integer coordinates are retrieved using bilinear interpolation. Every channel of multi-channel images is processed independently. Also the image should be a single channel or three channel image. While the center of the rectangle must be inside the image, parts of the rectangle may be outside.Parameters:image - Source image.patchSize - Size of the extracted patch.center - Floating point coordinates of the center of the extracted rectangle within the source image. The center must be inside the image.patch - Extracted patch that has the size patchSize and the same number of channels as src . SEE: warpAffine, warpPerspective

#### getRectSubPix public static void getRectSubPix([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Size](http://docs.google.com/org/opencv/core/Size.html) patchSize, [Point](http://docs.google.com/org/opencv/core/Point.html) center, [Mat](http://docs.google.com/org/opencv/core/Mat.html) patch, int patchType) Retrieves a pixel rectangle from an image with sub-pixel accuracy. The function getRectSubPix extracts pixels from src: \(patch(x, y) = src(x + \texttt{center.x} - ( \texttt{dst.cols} -1)\*0.5, y + \texttt{center.y} - ( \texttt{dst.rows} -1)\*0.5)\) where the values of the pixels at non-integer coordinates are retrieved using bilinear interpolation. Every channel of multi-channel images is processed independently. Also the image should be a single channel or three channel image. While the center of the rectangle must be inside the image, parts of the rectangle may be outside.Parameters:image - Source image.patchSize - Size of the extracted patch.center - Floating point coordinates of the center of the extracted rectangle within the source image. The center must be inside the image.patch - Extracted patch that has the size patchSize and the same number of channels as src .patchType - Depth of the extracted pixels. By default, they have the same depth as src . SEE: warpAffine, warpPerspective

#### getRotationMatrix2D public static [Mat](http://docs.google.com/org/opencv/core/Mat.html) getRotationMatrix2D([Point](http://docs.google.com/org/opencv/core/Point.html) center, double angle, double scale) Calculates an affine matrix of 2D rotation. The function calculates the following matrix: \(\begin{bmatrix} \alpha & \beta & (1- \alpha ) \cdot \texttt{center.x} - \beta \cdot \texttt{center.y} \\ - \beta & \alpha & \beta \cdot \texttt{center.x} + (1- \alpha ) \cdot \texttt{center.y} \end{bmatrix}\) where \(\begin{array}{l} \alpha = \texttt{scale} \cdot \cos \texttt{angle} , \\ \beta = \texttt{scale} \cdot \sin \texttt{angle} \end{array}\) The transformation maps the rotation center to itself. If this is not the target, adjust the shift.Parameters:center - Center of the rotation in the source image.angle - Rotation angle in degrees. Positive values mean counter-clockwise rotation (the coordinate origin is assumed to be the top-left corner).scale - Isotropic scale factor. SEE: getAffineTransform, warpAffine, transform Returns:automatically generated

#### getStructuringElement public static [Mat](http://docs.google.com/org/opencv/core/Mat.html) getStructuringElement(int shape, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize) Returns a structuring element of the specified size and shape for morphological operations. The function constructs and returns the structuring element that can be further passed to #erode, #dilate or #morphologyEx. But you can also construct an arbitrary binary mask yourself and use it as the structuring element.Parameters:shape - Element shape that could be one of #MorphShapesksize - Size of the structuring element. anchor is at the center. Note that only the shape of a cross-shaped element depends on the anchor position. In other cases the anchor just regulates how much the result of the morphological operation is shifted. Returns:automatically generated

#### getStructuringElement public static [Mat](http://docs.google.com/org/opencv/core/Mat.html) getStructuringElement(int shape, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor) Returns a structuring element of the specified size and shape for morphological operations. The function constructs and returns the structuring element that can be further passed to #erode, #dilate or #morphologyEx. But you can also construct an arbitrary binary mask yourself and use it as the structuring element.Parameters:shape - Element shape that could be one of #MorphShapesksize - Size of the structuring element.anchor - Anchor position within the element. The default value \((-1, -1)\) means that the anchor is at the center. Note that only the shape of a cross-shaped element depends on the anchor position. In other cases the anchor just regulates how much the result of the morphological operation is shifted. Returns:automatically generated

#### getTextSize public static [Size](http://docs.google.com/org/opencv/core/Size.html) getTextSize(java.lang.String text, int fontFace, double fontScale, int thickness, int[] baseLine)

#### goodFeaturesToTrack public static void goodFeaturesToTrack([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) corners, int maxCorners, double qualityLevel, double minDistance)

Determines strong corners on an image. The function finds the most prominent corners in the image or in the specified image region, as described in CITE: Shi94

* + - * Function calculates the corner quality measure at every source image pixel using the #cornerMinEigenVal or #cornerHarris .
      * Function performs a non-maximum suppression (the local maximums in \*3 x 3\* neighborhood are retained).
      * The corners with the minimal eigenvalue less than \(\texttt{qualityLevel} \cdot \max\_{x,y} qualityMeasureMap(x,y)\) are rejected.
      * The remaining corners are sorted by the quality measure in the descending order.
      * Function throws away each corner for which there is a stronger corner at a distance less than maxDistance.

The function can be used to initialize a point-based tracker of an object. **Note:** If the function is called with different values A and B of the parameter qualityLevel , and A > B, the vector of returned corners with qualityLevel=A will be the prefix of the output vector with qualityLevel=B .Parameters:image - Input 8-bit or floating-point 32-bit, single-channel image.corners - Output vector of detected corners.maxCorners - Maximum number of corners to return. If there are more corners than are found, the strongest of them is returned. maxCorners <= 0 implies that no limit on the maximum is set and all detected corners are returned.qualityLevel - Parameter characterizing the minimal accepted quality of image corners. The parameter value is multiplied by the best corner quality measure, which is the minimal eigenvalue (see #cornerMinEigenVal ) or the Harris function response (see #cornerHarris ). The corners with the quality measure less than the product are rejected. For example, if the best corner has the quality measure = 1500, and the qualityLevel=0.01 , then all the corners with the quality measure less than 15 are rejected.minDistance - Minimum possible Euclidean distance between the returned corners. CV\_8UC1 and the same size as image ), it specifies the region in which the corners are detected. pixel neighborhood. See cornerEigenValsAndVecs . or #cornerMinEigenVal. SEE: cornerMinEigenVal, cornerHarris, calcOpticalFlowPyrLK, estimateRigidTransform,

#### goodFeaturesToTrack public static void goodFeaturesToTrack([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) corners, int maxCorners, double qualityLevel, double minDistance, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask)

Determines strong corners on an image. The function finds the most prominent corners in the image or in the specified image region, as described in CITE: Shi94

* + - * Function calculates the corner quality measure at every source image pixel using the #cornerMinEigenVal or #cornerHarris .
      * Function performs a non-maximum suppression (the local maximums in \*3 x 3\* neighborhood are retained).
      * The corners with the minimal eigenvalue less than \(\texttt{qualityLevel} \cdot \max\_{x,y} qualityMeasureMap(x,y)\) are rejected.
      * The remaining corners are sorted by the quality measure in the descending order.
      * Function throws away each corner for which there is a stronger corner at a distance less than maxDistance.

The function can be used to initialize a point-based tracker of an object. **Note:** If the function is called with different values A and B of the parameter qualityLevel , and A > B, the vector of returned corners with qualityLevel=A will be the prefix of the output vector with qualityLevel=B .Parameters:image - Input 8-bit or floating-point 32-bit, single-channel image.corners - Output vector of detected corners.maxCorners - Maximum number of corners to return. If there are more corners than are found, the strongest of them is returned. maxCorners <= 0 implies that no limit on the maximum is set and all detected corners are returned.qualityLevel - Parameter characterizing the minimal accepted quality of image corners. The parameter value is multiplied by the best corner quality measure, which is the minimal eigenvalue (see #cornerMinEigenVal ) or the Harris function response (see #cornerHarris ). The corners with the quality measure less than the product are rejected. For example, if the best corner has the quality measure = 1500, and the qualityLevel=0.01 , then all the corners with the quality measure less than 15 are rejected.minDistance - Minimum possible Euclidean distance between the returned corners.mask - Optional region of interest. If the image is not empty (it needs to have the type CV\_8UC1 and the same size as image ), it specifies the region in which the corners are detected. pixel neighborhood. See cornerEigenValsAndVecs . or #cornerMinEigenVal. SEE: cornerMinEigenVal, cornerHarris, calcOpticalFlowPyrLK, estimateRigidTransform,

#### goodFeaturesToTrack public static void goodFeaturesToTrack([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) corners, int maxCorners, double qualityLevel, double minDistance, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, int blockSize)

Determines strong corners on an image. The function finds the most prominent corners in the image or in the specified image region, as described in CITE: Shi94

* + - * Function calculates the corner quality measure at every source image pixel using the #cornerMinEigenVal or #cornerHarris .
      * Function performs a non-maximum suppression (the local maximums in \*3 x 3\* neighborhood are retained).
      * The corners with the minimal eigenvalue less than \(\texttt{qualityLevel} \cdot \max\_{x,y} qualityMeasureMap(x,y)\) are rejected.
      * The remaining corners are sorted by the quality measure in the descending order.
      * Function throws away each corner for which there is a stronger corner at a distance less than maxDistance.

The function can be used to initialize a point-based tracker of an object. **Note:** If the function is called with different values A and B of the parameter qualityLevel , and A > B, the vector of returned corners with qualityLevel=A will be the prefix of the output vector with qualityLevel=B .Parameters:image - Input 8-bit or floating-point 32-bit, single-channel image.corners - Output vector of detected corners.maxCorners - Maximum number of corners to return. If there are more corners than are found, the strongest of them is returned. maxCorners <= 0 implies that no limit on the maximum is set and all detected corners are returned.qualityLevel - Parameter characterizing the minimal accepted quality of image corners. The parameter value is multiplied by the best corner quality measure, which is the minimal eigenvalue (see #cornerMinEigenVal ) or the Harris function response (see #cornerHarris ). The corners with the quality measure less than the product are rejected. For example, if the best corner has the quality measure = 1500, and the qualityLevel=0.01 , then all the corners with the quality measure less than 15 are rejected.minDistance - Minimum possible Euclidean distance between the returned corners.mask - Optional region of interest. If the image is not empty (it needs to have the type CV\_8UC1 and the same size as image ), it specifies the region in which the corners are detected.blockSize - Size of an average block for computing a derivative covariation matrix over each pixel neighborhood. See cornerEigenValsAndVecs . or #cornerMinEigenVal. SEE: cornerMinEigenVal, cornerHarris, calcOpticalFlowPyrLK, estimateRigidTransform,

#### goodFeaturesToTrack public static void goodFeaturesToTrack([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) corners, int maxCorners, double qualityLevel, double minDistance, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, int blockSize, boolean useHarrisDetector)

Determines strong corners on an image. The function finds the most prominent corners in the image or in the specified image region, as described in CITE: Shi94

* + - * Function calculates the corner quality measure at every source image pixel using the #cornerMinEigenVal or #cornerHarris .
      * Function performs a non-maximum suppression (the local maximums in \*3 x 3\* neighborhood are retained).
      * The corners with the minimal eigenvalue less than \(\texttt{qualityLevel} \cdot \max\_{x,y} qualityMeasureMap(x,y)\) are rejected.
      * The remaining corners are sorted by the quality measure in the descending order.
      * Function throws away each corner for which there is a stronger corner at a distance less than maxDistance.

The function can be used to initialize a point-based tracker of an object. **Note:** If the function is called with different values A and B of the parameter qualityLevel , and A > B, the vector of returned corners with qualityLevel=A will be the prefix of the output vector with qualityLevel=B .Parameters:image - Input 8-bit or floating-point 32-bit, single-channel image.corners - Output vector of detected corners.maxCorners - Maximum number of corners to return. If there are more corners than are found, the strongest of them is returned. maxCorners <= 0 implies that no limit on the maximum is set and all detected corners are returned.qualityLevel - Parameter characterizing the minimal accepted quality of image corners. The parameter value is multiplied by the best corner quality measure, which is the minimal eigenvalue (see #cornerMinEigenVal ) or the Harris function response (see #cornerHarris ). The corners with the quality measure less than the product are rejected. For example, if the best corner has the quality measure = 1500, and the qualityLevel=0.01 , then all the corners with the quality measure less than 15 are rejected.minDistance - Minimum possible Euclidean distance between the returned corners.mask - Optional region of interest. If the image is not empty (it needs to have the type CV\_8UC1 and the same size as image ), it specifies the region in which the corners are detected.blockSize - Size of an average block for computing a derivative covariation matrix over each pixel neighborhood. See cornerEigenValsAndVecs .useHarrisDetector - Parameter indicating whether to use a Harris detector (see #cornerHarris) or #cornerMinEigenVal. SEE: cornerMinEigenVal, cornerHarris, calcOpticalFlowPyrLK, estimateRigidTransform,

#### goodFeaturesToTrack public static void goodFeaturesToTrack([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) corners, int maxCorners, double qualityLevel, double minDistance, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, int blockSize, boolean useHarrisDetector, double k)

Determines strong corners on an image. The function finds the most prominent corners in the image or in the specified image region, as described in CITE: Shi94

* + - * Function calculates the corner quality measure at every source image pixel using the #cornerMinEigenVal or #cornerHarris .
      * Function performs a non-maximum suppression (the local maximums in \*3 x 3\* neighborhood are retained).
      * The corners with the minimal eigenvalue less than \(\texttt{qualityLevel} \cdot \max\_{x,y} qualityMeasureMap(x,y)\) are rejected.
      * The remaining corners are sorted by the quality measure in the descending order.
      * Function throws away each corner for which there is a stronger corner at a distance less than maxDistance.

The function can be used to initialize a point-based tracker of an object. **Note:** If the function is called with different values A and B of the parameter qualityLevel , and A > B, the vector of returned corners with qualityLevel=A will be the prefix of the output vector with qualityLevel=B .Parameters:image - Input 8-bit or floating-point 32-bit, single-channel image.corners - Output vector of detected corners.maxCorners - Maximum number of corners to return. If there are more corners than are found, the strongest of them is returned. maxCorners <= 0 implies that no limit on the maximum is set and all detected corners are returned.qualityLevel - Parameter characterizing the minimal accepted quality of image corners. The parameter value is multiplied by the best corner quality measure, which is the minimal eigenvalue (see #cornerMinEigenVal ) or the Harris function response (see #cornerHarris ). The corners with the quality measure less than the product are rejected. For example, if the best corner has the quality measure = 1500, and the qualityLevel=0.01 , then all the corners with the quality measure less than 15 are rejected.minDistance - Minimum possible Euclidean distance between the returned corners.mask - Optional region of interest. If the image is not empty (it needs to have the type CV\_8UC1 and the same size as image ), it specifies the region in which the corners are detected.blockSize - Size of an average block for computing a derivative covariation matrix over each pixel neighborhood. See cornerEigenValsAndVecs .useHarrisDetector - Parameter indicating whether to use a Harris detector (see #cornerHarris) or #cornerMinEigenVal.k - Free parameter of the Harris detector. SEE: cornerMinEigenVal, cornerHarris, calcOpticalFlowPyrLK, estimateRigidTransform,

#### goodFeaturesToTrack public static void goodFeaturesToTrack([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) corners, int maxCorners, double qualityLevel, double minDistance, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, int blockSize, int gradientSize)

#### goodFeaturesToTrack public static void goodFeaturesToTrack([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) corners, int maxCorners, double qualityLevel, double minDistance, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, int blockSize, int gradientSize, boolean useHarrisDetector)

#### goodFeaturesToTrack public static void goodFeaturesToTrack([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) corners, int maxCorners, double qualityLevel, double minDistance, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, int blockSize, int gradientSize, boolean useHarrisDetector, double k)

#### grabCut public static void grabCut([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Rect](http://docs.google.com/org/opencv/core/Rect.html) rect, [Mat](http://docs.google.com/org/opencv/core/Mat.html) bgdModel, [Mat](http://docs.google.com/org/opencv/core/Mat.html) fgdModel, int iterCount) Runs the GrabCut algorithm. The function implements the [GrabCut image segmentation algorithm](http://en.wikipedia.org/wiki/GrabCut).Parameters:img - Input 8-bit 3-channel image.mask - Input/output 8-bit single-channel mask. The mask is initialized by the function when mode is set to #GC\_INIT\_WITH\_RECT. Its elements may have one of the #GrabCutClasses.rect - ROI containing a segmented object. The pixels outside of the ROI are marked as "obvious background". The parameter is only used when mode==#GC\_INIT\_WITH\_RECT .bgdModel - Temporary array for the background model. Do not modify it while you are processing the same image.fgdModel - Temporary arrays for the foreground model. Do not modify it while you are processing the same image.iterCount - Number of iterations the algorithm should make before returning the result. Note that the result can be refined with further calls with mode==#GC\_INIT\_WITH\_MASK or mode==GC\_EVAL .

#### grabCut public static void grabCut([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Rect](http://docs.google.com/org/opencv/core/Rect.html) rect, [Mat](http://docs.google.com/org/opencv/core/Mat.html) bgdModel, [Mat](http://docs.google.com/org/opencv/core/Mat.html) fgdModel, int iterCount, int mode) Runs the GrabCut algorithm. The function implements the [GrabCut image segmentation algorithm](http://en.wikipedia.org/wiki/GrabCut).Parameters:img - Input 8-bit 3-channel image.mask - Input/output 8-bit single-channel mask. The mask is initialized by the function when mode is set to #GC\_INIT\_WITH\_RECT. Its elements may have one of the #GrabCutClasses.rect - ROI containing a segmented object. The pixels outside of the ROI are marked as "obvious background". The parameter is only used when mode==#GC\_INIT\_WITH\_RECT .bgdModel - Temporary array for the background model. Do not modify it while you are processing the same image.fgdModel - Temporary arrays for the foreground model. Do not modify it while you are processing the same image.iterCount - Number of iterations the algorithm should make before returning the result. Note that the result can be refined with further calls with mode==#GC\_INIT\_WITH\_MASK or mode==GC\_EVAL .mode - Operation mode that could be one of the #GrabCutModes

#### HoughCircles public static void HoughCircles([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) circles, int method, double dp, double minDist) Finds circles in a grayscale image using the Hough transform. The function finds circles in a grayscale image using a modification of the Hough transform. Example: : INCLUDE: snippets/imgproc\_HoughLinesCircles.cpp **Note:** Usually the function detects the centers of circles well. However, it may fail to find correct radii. You can assist to the function by specifying the radius range ( minRadius and maxRadius ) if you know it. Or, you may set maxRadius to a negative number to return centers only without radius search, and find the correct radius using an additional procedure.Parameters:image - 8-bit, single-channel, grayscale input image.circles - Output vector of found circles. Each vector is encoded as 3 or 4 element floating-point vector \((x, y, radius)\) or \((x, y, radius, votes)\) .method - Detection method, see #HoughModes. Currently, the only implemented method is #HOUGH\_GRADIENTdp - Inverse ratio of the accumulator resolution to the image resolution. For example, if dp=1 , the accumulator has the same resolution as the input image. If dp=2 , the accumulator has half as big width and height.minDist - Minimum distance between the centers of the detected circles. If the parameter is too small, multiple neighbor circles may be falsely detected in addition to a true one. If it is too large, some circles may be missed. threshold of the two passed to the Canny edge detector (the lower one is twice smaller). accumulator threshold for the circle centers at the detection stage. The smaller it is, the more false circles may be detected. Circles, corresponding to the larger accumulator values, will be returned first. centers without finding the radius. SEE: fitEllipse, minEnclosingCircle

#### HoughCircles public static void HoughCircles([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) circles, int method, double dp, double minDist, double param1) Finds circles in a grayscale image using the Hough transform. The function finds circles in a grayscale image using a modification of the Hough transform. Example: : INCLUDE: snippets/imgproc\_HoughLinesCircles.cpp **Note:** Usually the function detects the centers of circles well. However, it may fail to find correct radii. You can assist to the function by specifying the radius range ( minRadius and maxRadius ) if you know it. Or, you may set maxRadius to a negative number to return centers only without radius search, and find the correct radius using an additional procedure.Parameters:image - 8-bit, single-channel, grayscale input image.circles - Output vector of found circles. Each vector is encoded as 3 or 4 element floating-point vector \((x, y, radius)\) or \((x, y, radius, votes)\) .method - Detection method, see #HoughModes. Currently, the only implemented method is #HOUGH\_GRADIENTdp - Inverse ratio of the accumulator resolution to the image resolution. For example, if dp=1 , the accumulator has the same resolution as the input image. If dp=2 , the accumulator has half as big width and height.minDist - Minimum distance between the centers of the detected circles. If the parameter is too small, multiple neighbor circles may be falsely detected in addition to a true one. If it is too large, some circles may be missed.param1 - First method-specific parameter. In case of #HOUGH\_GRADIENT , it is the higher threshold of the two passed to the Canny edge detector (the lower one is twice smaller). accumulator threshold for the circle centers at the detection stage. The smaller it is, the more false circles may be detected. Circles, corresponding to the larger accumulator values, will be returned first. centers without finding the radius. SEE: fitEllipse, minEnclosingCircle

#### HoughCircles public static void HoughCircles([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) circles, int method, double dp, double minDist, double param1, double param2) Finds circles in a grayscale image using the Hough transform. The function finds circles in a grayscale image using a modification of the Hough transform. Example: : INCLUDE: snippets/imgproc\_HoughLinesCircles.cpp **Note:** Usually the function detects the centers of circles well. However, it may fail to find correct radii. You can assist to the function by specifying the radius range ( minRadius and maxRadius ) if you know it. Or, you may set maxRadius to a negative number to return centers only without radius search, and find the correct radius using an additional procedure.Parameters:image - 8-bit, single-channel, grayscale input image.circles - Output vector of found circles. Each vector is encoded as 3 or 4 element floating-point vector \((x, y, radius)\) or \((x, y, radius, votes)\) .method - Detection method, see #HoughModes. Currently, the only implemented method is #HOUGH\_GRADIENTdp - Inverse ratio of the accumulator resolution to the image resolution. For example, if dp=1 , the accumulator has the same resolution as the input image. If dp=2 , the accumulator has half as big width and height.minDist - Minimum distance between the centers of the detected circles. If the parameter is too small, multiple neighbor circles may be falsely detected in addition to a true one. If it is too large, some circles may be missed.param1 - First method-specific parameter. In case of #HOUGH\_GRADIENT , it is the higher threshold of the two passed to the Canny edge detector (the lower one is twice smaller).param2 - Second method-specific parameter. In case of #HOUGH\_GRADIENT , it is the accumulator threshold for the circle centers at the detection stage. The smaller it is, the more false circles may be detected. Circles, corresponding to the larger accumulator values, will be returned first. centers without finding the radius. SEE: fitEllipse, minEnclosingCircle

#### HoughCircles public static void HoughCircles([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) circles, int method, double dp, double minDist, double param1, double param2, int minRadius) Finds circles in a grayscale image using the Hough transform. The function finds circles in a grayscale image using a modification of the Hough transform. Example: : INCLUDE: snippets/imgproc\_HoughLinesCircles.cpp **Note:** Usually the function detects the centers of circles well. However, it may fail to find correct radii. You can assist to the function by specifying the radius range ( minRadius and maxRadius ) if you know it. Or, you may set maxRadius to a negative number to return centers only without radius search, and find the correct radius using an additional procedure.Parameters:image - 8-bit, single-channel, grayscale input image.circles - Output vector of found circles. Each vector is encoded as 3 or 4 element floating-point vector \((x, y, radius)\) or \((x, y, radius, votes)\) .method - Detection method, see #HoughModes. Currently, the only implemented method is #HOUGH\_GRADIENTdp - Inverse ratio of the accumulator resolution to the image resolution. For example, if dp=1 , the accumulator has the same resolution as the input image. If dp=2 , the accumulator has half as big width and height.minDist - Minimum distance between the centers of the detected circles. If the parameter is too small, multiple neighbor circles may be falsely detected in addition to a true one. If it is too large, some circles may be missed.param1 - First method-specific parameter. In case of #HOUGH\_GRADIENT , it is the higher threshold of the two passed to the Canny edge detector (the lower one is twice smaller).param2 - Second method-specific parameter. In case of #HOUGH\_GRADIENT , it is the accumulator threshold for the circle centers at the detection stage. The smaller it is, the more false circles may be detected. Circles, corresponding to the larger accumulator values, will be returned first.minRadius - Minimum circle radius. centers without finding the radius. SEE: fitEllipse, minEnclosingCircle

#### HoughCircles public static void HoughCircles([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) circles, int method, double dp, double minDist, double param1, double param2, int minRadius, int maxRadius) Finds circles in a grayscale image using the Hough transform. The function finds circles in a grayscale image using a modification of the Hough transform. Example: : INCLUDE: snippets/imgproc\_HoughLinesCircles.cpp **Note:** Usually the function detects the centers of circles well. However, it may fail to find correct radii. You can assist to the function by specifying the radius range ( minRadius and maxRadius ) if you know it. Or, you may set maxRadius to a negative number to return centers only without radius search, and find the correct radius using an additional procedure.Parameters:image - 8-bit, single-channel, grayscale input image.circles - Output vector of found circles. Each vector is encoded as 3 or 4 element floating-point vector \((x, y, radius)\) or \((x, y, radius, votes)\) .method - Detection method, see #HoughModes. Currently, the only implemented method is #HOUGH\_GRADIENTdp - Inverse ratio of the accumulator resolution to the image resolution. For example, if dp=1 , the accumulator has the same resolution as the input image. If dp=2 , the accumulator has half as big width and height.minDist - Minimum distance between the centers of the detected circles. If the parameter is too small, multiple neighbor circles may be falsely detected in addition to a true one. If it is too large, some circles may be missed.param1 - First method-specific parameter. In case of #HOUGH\_GRADIENT , it is the higher threshold of the two passed to the Canny edge detector (the lower one is twice smaller).param2 - Second method-specific parameter. In case of #HOUGH\_GRADIENT , it is the accumulator threshold for the circle centers at the detection stage. The smaller it is, the more false circles may be detected. Circles, corresponding to the larger accumulator values, will be returned first.minRadius - Minimum circle radius.maxRadius - Maximum circle radius. If <= 0, uses the maximum image dimension. If < 0, returns centers without finding the radius. SEE: fitEllipse, minEnclosingCircle

#### HoughLines public static void HoughLines([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines, double rho, double theta, int threshold) Finds lines in a binary image using the standard Hough transform. The function implements the standard or standard multi-scale Hough transform algorithm for line detection. See <http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm> for a good explanation of Hough transform.Parameters:image - 8-bit, single-channel binary source image. The image may be modified by the function.lines - Output vector of lines. Each line is represented by a 2 or 3 element vector \((\rho, \theta)\) or \((\rho, \theta, \textrm{votes})\) . \(\rho\) is the distance from the coordinate origin \((0,0)\) (top-left corner of the image). \(\theta\) is the line rotation angle in radians ( \(0 \sim \textrm{vertical line}, \pi/2 \sim \textrm{horizontal line}\) ). \(\textrm{votes}\) is the value of accumulator.rho - Distance resolution of the accumulator in pixels.theta - Angle resolution of the accumulator in radians.threshold - Accumulator threshold parameter. Only those lines are returned that get enough votes ( \(>\texttt{threshold}\) ). The coarse accumulator distance resolution is rho and the accurate accumulator resolution is rho/srn . If both srn=0 and stn=0 , the classical Hough transform is used. Otherwise, both these parameters should be positive. Must fall between 0 and max\_theta. Must fall between min\_theta and CV\_PI.

#### HoughLines public static void HoughLines([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines, double rho, double theta, int threshold, double srn) Finds lines in a binary image using the standard Hough transform. The function implements the standard or standard multi-scale Hough transform algorithm for line detection. See <http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm> for a good explanation of Hough transform.Parameters:image - 8-bit, single-channel binary source image. The image may be modified by the function.lines - Output vector of lines. Each line is represented by a 2 or 3 element vector \((\rho, \theta)\) or \((\rho, \theta, \textrm{votes})\) . \(\rho\) is the distance from the coordinate origin \((0,0)\) (top-left corner of the image). \(\theta\) is the line rotation angle in radians ( \(0 \sim \textrm{vertical line}, \pi/2 \sim \textrm{horizontal line}\) ). \(\textrm{votes}\) is the value of accumulator.rho - Distance resolution of the accumulator in pixels.theta - Angle resolution of the accumulator in radians.threshold - Accumulator threshold parameter. Only those lines are returned that get enough votes ( \(>\texttt{threshold}\) ).srn - For the multi-scale Hough transform, it is a divisor for the distance resolution rho . The coarse accumulator distance resolution is rho and the accurate accumulator resolution is rho/srn . If both srn=0 and stn=0 , the classical Hough transform is used. Otherwise, both these parameters should be positive. Must fall between 0 and max\_theta. Must fall between min\_theta and CV\_PI.

#### HoughLines public static void HoughLines([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines, double rho, double theta, int threshold, double srn, double stn) Finds lines in a binary image using the standard Hough transform. The function implements the standard or standard multi-scale Hough transform algorithm for line detection. See <http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm> for a good explanation of Hough transform.Parameters:image - 8-bit, single-channel binary source image. The image may be modified by the function.lines - Output vector of lines. Each line is represented by a 2 or 3 element vector \((\rho, \theta)\) or \((\rho, \theta, \textrm{votes})\) . \(\rho\) is the distance from the coordinate origin \((0,0)\) (top-left corner of the image). \(\theta\) is the line rotation angle in radians ( \(0 \sim \textrm{vertical line}, \pi/2 \sim \textrm{horizontal line}\) ). \(\textrm{votes}\) is the value of accumulator.rho - Distance resolution of the accumulator in pixels.theta - Angle resolution of the accumulator in radians.threshold - Accumulator threshold parameter. Only those lines are returned that get enough votes ( \(>\texttt{threshold}\) ).srn - For the multi-scale Hough transform, it is a divisor for the distance resolution rho . The coarse accumulator distance resolution is rho and the accurate accumulator resolution is rho/srn . If both srn=0 and stn=0 , the classical Hough transform is used. Otherwise, both these parameters should be positive.stn - For the multi-scale Hough transform, it is a divisor for the distance resolution theta. Must fall between 0 and max\_theta. Must fall between min\_theta and CV\_PI.

#### HoughLines public static void HoughLines([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines, double rho, double theta, int threshold, double srn, double stn, double min\_theta) Finds lines in a binary image using the standard Hough transform. The function implements the standard or standard multi-scale Hough transform algorithm for line detection. See <http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm> for a good explanation of Hough transform.Parameters:image - 8-bit, single-channel binary source image. The image may be modified by the function.lines - Output vector of lines. Each line is represented by a 2 or 3 element vector \((\rho, \theta)\) or \((\rho, \theta, \textrm{votes})\) . \(\rho\) is the distance from the coordinate origin \((0,0)\) (top-left corner of the image). \(\theta\) is the line rotation angle in radians ( \(0 \sim \textrm{vertical line}, \pi/2 \sim \textrm{horizontal line}\) ). \(\textrm{votes}\) is the value of accumulator.rho - Distance resolution of the accumulator in pixels.theta - Angle resolution of the accumulator in radians.threshold - Accumulator threshold parameter. Only those lines are returned that get enough votes ( \(>\texttt{threshold}\) ).srn - For the multi-scale Hough transform, it is a divisor for the distance resolution rho . The coarse accumulator distance resolution is rho and the accurate accumulator resolution is rho/srn . If both srn=0 and stn=0 , the classical Hough transform is used. Otherwise, both these parameters should be positive.stn - For the multi-scale Hough transform, it is a divisor for the distance resolution theta.min\_theta - For standard and multi-scale Hough transform, minimum angle to check for lines. Must fall between 0 and max\_theta. Must fall between min\_theta and CV\_PI.

#### HoughLines public static void HoughLines([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines, double rho, double theta, int threshold, double srn, double stn, double min\_theta, double max\_theta) Finds lines in a binary image using the standard Hough transform. The function implements the standard or standard multi-scale Hough transform algorithm for line detection. See <http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm> for a good explanation of Hough transform.Parameters:image - 8-bit, single-channel binary source image. The image may be modified by the function.lines - Output vector of lines. Each line is represented by a 2 or 3 element vector \((\rho, \theta)\) or \((\rho, \theta, \textrm{votes})\) . \(\rho\) is the distance from the coordinate origin \((0,0)\) (top-left corner of the image). \(\theta\) is the line rotation angle in radians ( \(0 \sim \textrm{vertical line}, \pi/2 \sim \textrm{horizontal line}\) ). \(\textrm{votes}\) is the value of accumulator.rho - Distance resolution of the accumulator in pixels.theta - Angle resolution of the accumulator in radians.threshold - Accumulator threshold parameter. Only those lines are returned that get enough votes ( \(>\texttt{threshold}\) ).srn - For the multi-scale Hough transform, it is a divisor for the distance resolution rho . The coarse accumulator distance resolution is rho and the accurate accumulator resolution is rho/srn . If both srn=0 and stn=0 , the classical Hough transform is used. Otherwise, both these parameters should be positive.stn - For the multi-scale Hough transform, it is a divisor for the distance resolution theta.min\_theta - For standard and multi-scale Hough transform, minimum angle to check for lines. Must fall between 0 and max\_theta.max\_theta - For standard and multi-scale Hough transform, maximum angle to check for lines. Must fall between min\_theta and CV\_PI.

#### HoughLinesP public static void HoughLinesP([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines, double rho, double theta, int threshold) Finds line segments in a binary image using the probabilistic Hough transform. The function implements the probabilistic Hough transform algorithm for line detection, described in CITE: Matas00 See the line detection example below: INCLUDE: snippets/imgproc\_HoughLinesP.cpp This is a sample picture the function parameters have been tuned for: ![image](pics/building.jpg) And this is the output of the above program in case of the probabilistic Hough transform: ![image](pics/houghp.png)Parameters:image - 8-bit, single-channel binary source image. The image may be modified by the function.lines - Output vector of lines. Each line is represented by a 4-element vector \((x\_1, y\_1, x\_2, y\_2)\) , where \((x\_1,y\_1)\) and \((x\_2, y\_2)\) are the ending points of each detected line segment.rho - Distance resolution of the accumulator in pixels.theta - Angle resolution of the accumulator in radians.threshold - Accumulator threshold parameter. Only those lines are returned that get enough votes ( \(>\texttt{threshold}\) ). SEE: LineSegmentDetector

#### HoughLinesP public static void HoughLinesP([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines, double rho, double theta, int threshold, double minLineLength) Finds line segments in a binary image using the probabilistic Hough transform. The function implements the probabilistic Hough transform algorithm for line detection, described in CITE: Matas00 See the line detection example below: INCLUDE: snippets/imgproc\_HoughLinesP.cpp This is a sample picture the function parameters have been tuned for: ![image](pics/building.jpg) And this is the output of the above program in case of the probabilistic Hough transform: ![image](pics/houghp.png)Parameters:image - 8-bit, single-channel binary source image. The image may be modified by the function.lines - Output vector of lines. Each line is represented by a 4-element vector \((x\_1, y\_1, x\_2, y\_2)\) , where \((x\_1,y\_1)\) and \((x\_2, y\_2)\) are the ending points of each detected line segment.rho - Distance resolution of the accumulator in pixels.theta - Angle resolution of the accumulator in radians.threshold - Accumulator threshold parameter. Only those lines are returned that get enough votes ( \(>\texttt{threshold}\) ).minLineLength - Minimum line length. Line segments shorter than that are rejected. SEE: LineSegmentDetector

#### HoughLinesP public static void HoughLinesP([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines, double rho, double theta, int threshold, double minLineLength, double maxLineGap) Finds line segments in a binary image using the probabilistic Hough transform. The function implements the probabilistic Hough transform algorithm for line detection, described in CITE: Matas00 See the line detection example below: INCLUDE: snippets/imgproc\_HoughLinesP.cpp This is a sample picture the function parameters have been tuned for: ![image](pics/building.jpg) And this is the output of the above program in case of the probabilistic Hough transform: ![image](pics/houghp.png)Parameters:image - 8-bit, single-channel binary source image. The image may be modified by the function.lines - Output vector of lines. Each line is represented by a 4-element vector \((x\_1, y\_1, x\_2, y\_2)\) , where \((x\_1,y\_1)\) and \((x\_2, y\_2)\) are the ending points of each detected line segment.rho - Distance resolution of the accumulator in pixels.theta - Angle resolution of the accumulator in radians.threshold - Accumulator threshold parameter. Only those lines are returned that get enough votes ( \(>\texttt{threshold}\) ).minLineLength - Minimum line length. Line segments shorter than that are rejected.maxLineGap - Maximum allowed gap between points on the same line to link them. SEE: LineSegmentDetector

#### HoughLinesPointSet public static void HoughLinesPointSet([Mat](http://docs.google.com/org/opencv/core/Mat.html) \_point, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_lines, int lines\_max, int threshold, double min\_rho, double max\_rho, double rho\_step, double min\_theta, double max\_theta, double theta\_step) Finds lines in a set of points using the standard Hough transform. The function finds lines in a set of points using a modification of the Hough transform. INCLUDE: snippets/imgproc\_HoughLinesPointSet.cppParameters:\_point - Input vector of points. Each vector must be encoded as a Point vector \((x,y)\). Type must be CV\_32FC2 or CV\_32SC2.\_lines - Output vector of found lines. Each vector is encoded as a vector<Vec3d> \((votes, rho, theta)\). The larger the value of 'votes', the higher the reliability of the Hough line.lines\_max - Max count of hough lines.threshold - Accumulator threshold parameter. Only those lines are returned that get enough votes ( \(>\texttt{threshold}\) )min\_rho - Minimum Distance value of the accumulator in pixels.max\_rho - Maximum Distance value of the accumulator in pixels.rho\_step - Distance resolution of the accumulator in pixels.min\_theta - Minimum angle value of the accumulator in radians.max\_theta - Maximum angle value of the accumulator in radians.theta\_step - Angle resolution of the accumulator in radians.

#### HuMoments public static void HuMoments([Moments](http://docs.google.com/org/opencv/imgproc/Moments.html) m, [Mat](http://docs.google.com/org/opencv/core/Mat.html) hu)

#### initUndistortRectifyMap public static void initUndistortRectifyMap([Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) R, [Mat](http://docs.google.com/org/opencv/core/Mat.html) newCameraMatrix, [Size](http://docs.google.com/org/opencv/core/Size.html) size, int m1type, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map2) Computes the undistortion and rectification transformation map. The function computes the joint undistortion and rectification transformation and represents the result in the form of maps for remap. The undistorted image looks like original, as if it is captured with a camera using the camera matrix =newCameraMatrix and zero distortion. In case of a monocular camera, newCameraMatrix is usually equal to cameraMatrix, or it can be computed by #getOptimalNewCameraMatrix for a better control over scaling. In case of a stereo camera, newCameraMatrix is normally set to P1 or P2 computed by #stereoRectify . Also, this new camera is oriented differently in the coordinate space, according to R. That, for example, helps to align two heads of a stereo camera so that the epipolar lines on both images become horizontal and have the same y- coordinate (in case of a horizontally aligned stereo camera). The function actually builds the maps for the inverse mapping algorithm that is used by remap. That is, for each pixel \((u, v)\) in the destination (corrected and rectified) image, the function computes the corresponding coordinates in the source image (that is, in the original image from camera). The following process is applied: \( \begin{array}{l} x \leftarrow (u - {c'}\_x)/{f'}\_x \\ y \leftarrow (v - {c'}\_y)/{f'}\_y \\ {[X\,Y\,W]} ^T \leftarrow R^{-1}\*[x \, y \, 1]^T \\ x' \leftarrow X/W \\ y' \leftarrow Y/W \\ r^2 \leftarrow x'^2 + y'^2 \\ x'' \leftarrow x' \frac{1 + k\_1 r^2 + k\_2 r^4 + k\_3 r^6}{1 + k\_4 r^2 + k\_5 r^4 + k\_6 r^6} + 2p\_1 x' y' + p\_2(r^2 + 2 x'^2) + s\_1 r^2 + s\_2 r^4\\ y'' \leftarrow y' \frac{1 + k\_1 r^2 + k\_2 r^4 + k\_3 r^6}{1 + k\_4 r^2 + k\_5 r^4 + k\_6 r^6} + p\_1 (r^2 + 2 y'^2) + 2 p\_2 x' y' + s\_3 r^2 + s\_4 r^4 \\ s\vecthree{x'''}{y'''}{1} = \vecthreethree{R\_{33}(\tau\_x, \tau\_y)}{0}{-R\_{13}((\tau\_x, \tau\_y)} {0}{R\_{33}(\tau\_x, \tau\_y)}{-R\_{23}(\tau\_x, \tau\_y)} {0}{0}{1} R(\tau\_x, \tau\_y) \vecthree{x''}{y''}{1}\\ map\_x(u,v) \leftarrow x''' f\_x + c\_x \\ map\_y(u,v) \leftarrow y''' f\_y + c\_y \end{array} \) where \((k\_1, k\_2, p\_1, p\_2[, k\_3[, k\_4, k\_5, k\_6[, s\_1, s\_2, s\_3, s\_4[, \tau\_x, \tau\_y]]]])\) are the distortion coefficients. In case of a stereo camera, this function is called twice: once for each camera head, after stereoRectify, which in its turn is called after #stereoCalibrate. But if the stereo camera was not calibrated, it is still possible to compute the rectification transformations directly from the fundamental matrix using #stereoRectifyUncalibrated. For each camera, the function computes homography H as the rectification transformation in a pixel domain, not a rotation matrix R in 3D space. R can be computed from H as \(\texttt{R} = \texttt{cameraMatrix} ^{-1} \cdot \texttt{H} \cdot \texttt{cameraMatrix}\) where cameraMatrix can be chosen arbitrarily.Parameters:cameraMatrix - Input camera matrix \(A=\vecthreethree{f\_x}{0}{c\_x}{0}{f\_y}{c\_y}{0}{0}{1}\) .distCoeffs - Input vector of distortion coefficients \((k\_1, k\_2, p\_1, p\_2[, k\_3[, k\_4, k\_5, k\_6[, s\_1, s\_2, s\_3, s\_4[, \tau\_x, \tau\_y]]]])\) of 4, 5, 8, 12 or 14 elements. If the vector is NULL/empty, the zero distortion coefficients are assumed.R - Optional rectification transformation in the object space (3x3 matrix). R1 or R2 , computed by #stereoRectify can be passed here. If the matrix is empty, the identity transformation is assumed. In cvInitUndistortMap R assumed to be an identity matrix.newCameraMatrix - New camera matrix \(A'=\vecthreethree{f\_x'}{0}{c\_x'}{0}{f\_y'}{c\_y'}{0}{0}{1}\).size - Undistorted image size.m1type - Type of the first output map that can be CV\_32FC1, CV\_32FC2 or CV\_16SC2, see #convertMapsmap1 - The first output map.map2 - The second output map.

#### initWideAngleProjMap public static float initWideAngleProjMap([Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs, [Size](http://docs.google.com/org/opencv/core/Size.html) imageSize, int destImageWidth, int m1type, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map2)

#### initWideAngleProjMap public static float initWideAngleProjMap([Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs, [Size](http://docs.google.com/org/opencv/core/Size.html) imageSize, int destImageWidth, int m1type, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map2, int projType)

#### initWideAngleProjMap public static float initWideAngleProjMap([Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs, [Size](http://docs.google.com/org/opencv/core/Size.html) imageSize, int destImageWidth, int m1type, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map2, int projType, double alpha)

#### integral public static void integral([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sum)

#### integral public static void integral([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sum, int sdepth)

#### integral2 public static void integral2([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sum, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sqsum)

#### integral2 public static void integral2([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sum, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sqsum, int sdepth)

#### integral2 public static void integral2([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sum, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sqsum, int sdepth, int sqdepth)

#### integral3 public static void integral3([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sum, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sqsum, [Mat](http://docs.google.com/org/opencv/core/Mat.html) tilted) Calculates the integral of an image. The function calculates one or more integral images for the source image as follows: \(\texttt{sum} (X,Y) = \sum \_{x<X,y<Y} \texttt{image} (x,y)\) \(\texttt{sqsum} (X,Y) = \sum \_{x<X,y<Y} \texttt{image} (x,y)^2\) \(\texttt{tilted} (X,Y) = \sum \_{y<Y,abs(x-X+1) \leq Y-y-1} \texttt{image} (x,y)\) Using these integral images, you can calculate sum, mean, and standard deviation over a specific up-right or rotated rectangular region of the image in a constant time, for example: \(\sum \_{x\_1 \leq x < x\_2, \, y\_1 \leq y < y\_2} \texttt{image} (x,y) = \texttt{sum} (x\_2,y\_2)- \texttt{sum} (x\_1,y\_2)- \texttt{sum} (x\_2,y\_1)+ \texttt{sum} (x\_1,y\_1)\) It makes possible to do a fast blurring or fast block correlation with a variable window size, for example. In case of multi-channel images, sums for each channel are accumulated independently. As a practical example, the next figure shows the calculation of the integral of a straight rectangle Rect(3,3,3,2) and of a tilted rectangle Rect(5,1,2,3) . The selected pixels in the original image are shown, as well as the relative pixels in the integral images sum and tilted . ![integral calculation example](pics/integral.png)Parameters:src - input image as \(W \times H\), 8-bit or floating-point (32f or 64f).sum - integral image as \((W+1)\times (H+1)\) , 32-bit integer or floating-point (32f or 64f).sqsum - integral image for squared pixel values; it is \((W+1)\times (H+1)\), double-precision floating-point (64f) array.tilted - integral for the image rotated by 45 degrees; it is \((W+1)\times (H+1)\) array with the same data type as sum. CV\_64F.

#### integral3 public static void integral3([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sum, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sqsum, [Mat](http://docs.google.com/org/opencv/core/Mat.html) tilted, int sdepth) Calculates the integral of an image. The function calculates one or more integral images for the source image as follows: \(\texttt{sum} (X,Y) = \sum \_{x<X,y<Y} \texttt{image} (x,y)\) \(\texttt{sqsum} (X,Y) = \sum \_{x<X,y<Y} \texttt{image} (x,y)^2\) \(\texttt{tilted} (X,Y) = \sum \_{y<Y,abs(x-X+1) \leq Y-y-1} \texttt{image} (x,y)\) Using these integral images, you can calculate sum, mean, and standard deviation over a specific up-right or rotated rectangular region of the image in a constant time, for example: \(\sum \_{x\_1 \leq x < x\_2, \, y\_1 \leq y < y\_2} \texttt{image} (x,y) = \texttt{sum} (x\_2,y\_2)- \texttt{sum} (x\_1,y\_2)- \texttt{sum} (x\_2,y\_1)+ \texttt{sum} (x\_1,y\_1)\) It makes possible to do a fast blurring or fast block correlation with a variable window size, for example. In case of multi-channel images, sums for each channel are accumulated independently. As a practical example, the next figure shows the calculation of the integral of a straight rectangle Rect(3,3,3,2) and of a tilted rectangle Rect(5,1,2,3) . The selected pixels in the original image are shown, as well as the relative pixels in the integral images sum and tilted . ![integral calculation example](pics/integral.png)Parameters:src - input image as \(W \times H\), 8-bit or floating-point (32f or 64f).sum - integral image as \((W+1)\times (H+1)\) , 32-bit integer or floating-point (32f or 64f).sqsum - integral image for squared pixel values; it is \((W+1)\times (H+1)\), double-precision floating-point (64f) array.tilted - integral for the image rotated by 45 degrees; it is \((W+1)\times (H+1)\) array with the same data type as sum.sdepth - desired depth of the integral and the tilted integral images, CV\_32S, CV\_32F, or CV\_64F.

#### integral3 public static void integral3([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sum, [Mat](http://docs.google.com/org/opencv/core/Mat.html) sqsum, [Mat](http://docs.google.com/org/opencv/core/Mat.html) tilted, int sdepth, int sqdepth) Calculates the integral of an image. The function calculates one or more integral images for the source image as follows: \(\texttt{sum} (X,Y) = \sum \_{x<X,y<Y} \texttt{image} (x,y)\) \(\texttt{sqsum} (X,Y) = \sum \_{x<X,y<Y} \texttt{image} (x,y)^2\) \(\texttt{tilted} (X,Y) = \sum \_{y<Y,abs(x-X+1) \leq Y-y-1} \texttt{image} (x,y)\) Using these integral images, you can calculate sum, mean, and standard deviation over a specific up-right or rotated rectangular region of the image in a constant time, for example: \(\sum \_{x\_1 \leq x < x\_2, \, y\_1 \leq y < y\_2} \texttt{image} (x,y) = \texttt{sum} (x\_2,y\_2)- \texttt{sum} (x\_1,y\_2)- \texttt{sum} (x\_2,y\_1)+ \texttt{sum} (x\_1,y\_1)\) It makes possible to do a fast blurring or fast block correlation with a variable window size, for example. In case of multi-channel images, sums for each channel are accumulated independently. As a practical example, the next figure shows the calculation of the integral of a straight rectangle Rect(3,3,3,2) and of a tilted rectangle Rect(5,1,2,3) . The selected pixels in the original image are shown, as well as the relative pixels in the integral images sum and tilted . ![integral calculation example](pics/integral.png)Parameters:src - input image as \(W \times H\), 8-bit or floating-point (32f or 64f).sum - integral image as \((W+1)\times (H+1)\) , 32-bit integer or floating-point (32f or 64f).sqsum - integral image for squared pixel values; it is \((W+1)\times (H+1)\), double-precision floating-point (64f) array.tilted - integral for the image rotated by 45 degrees; it is \((W+1)\times (H+1)\) array with the same data type as sum.sdepth - desired depth of the integral and the tilted integral images, CV\_32S, CV\_32F, or CV\_64F.sqdepth - desired depth of the integral image of squared pixel values, CV\_32F or CV\_64F.

#### intersectConvexConvex public static float intersectConvexConvex([Mat](http://docs.google.com/org/opencv/core/Mat.html) \_p1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_p2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_p12) Finds intersection of two convex polygonsParameters:\_p1 - First polygon\_p2 - Second polygon\_p12 - Output polygon describing the intersecting area When false, no intersection is found. If the polygons share a side or the vertex of one polygon lies on an edge of the other, they are not considered nested and an intersection will be found regardless of the value of handleNested. Returns:Absolute value of area of intersecting polygon **Note:** intersectConvexConvex doesn't confirm that both polygons are convex and will return invalid results if they aren't.

#### intersectConvexConvex public static float intersectConvexConvex([Mat](http://docs.google.com/org/opencv/core/Mat.html) \_p1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_p2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_p12, boolean handleNested) Finds intersection of two convex polygonsParameters:\_p1 - First polygon\_p2 - Second polygon\_p12 - Output polygon describing the intersecting areahandleNested - When true, an intersection is found if one of the polygons is fully enclosed in the other. When false, no intersection is found. If the polygons share a side or the vertex of one polygon lies on an edge of the other, they are not considered nested and an intersection will be found regardless of the value of handleNested. Returns:Absolute value of area of intersecting polygon **Note:** intersectConvexConvex doesn't confirm that both polygons are convex and will return invalid results if they aren't.

#### invertAffineTransform public static void invertAffineTransform([Mat](http://docs.google.com/org/opencv/core/Mat.html) M, [Mat](http://docs.google.com/org/opencv/core/Mat.html) iM) Inverts an affine transformation. The function computes an inverse affine transformation represented by \(2 \times 3\) matrix M: \(\begin{bmatrix} a\_{11} & a\_{12} & b\_1 \\ a\_{21} & a\_{22} & b\_2 \end{bmatrix}\) The result is also a \(2 \times 3\) matrix of the same type as M.Parameters:M - Original affine transformation.iM - Output reverse affine transformation.

#### isContourConvex public static boolean isContourConvex([MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html) contour) Tests a contour convexity. The function tests whether the input contour is convex or not. The contour must be simple, that is, without self-intersections. Otherwise, the function output is undefined.Parameters:contour - Input vector of 2D points, stored in std::vector<> or Mat Returns:automatically generated

#### Laplacian public static void Laplacian([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth) Calculates the Laplacian of an image. The function calculates the Laplacian of the source image by adding up the second x and y derivatives calculated using the Sobel operator: \(\texttt{dst} = \Delta \texttt{src} = \frac{\partial^2 \texttt{src}}{\partial x^2} + \frac{\partial^2 \texttt{src}}{\partial y^2}\) This is done when ksize > 1. When ksize == 1, the Laplacian is computed by filtering the image with the following \(3 \times 3\) aperture: \(\vecthreethree {0}{1}{0}{1}{-4}{1}{0}{1}{0}\)Parameters:src - Source image.dst - Destination image of the same size and the same number of channels as src .ddepth - Desired depth of the destination image. details. The size must be positive and odd. applied. See #getDerivKernels for details. SEE: Sobel, Scharr

#### Laplacian public static void Laplacian([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int ksize) Calculates the Laplacian of an image. The function calculates the Laplacian of the source image by adding up the second x and y derivatives calculated using the Sobel operator: \(\texttt{dst} = \Delta \texttt{src} = \frac{\partial^2 \texttt{src}}{\partial x^2} + \frac{\partial^2 \texttt{src}}{\partial y^2}\) This is done when ksize > 1. When ksize == 1, the Laplacian is computed by filtering the image with the following \(3 \times 3\) aperture: \(\vecthreethree {0}{1}{0}{1}{-4}{1}{0}{1}{0}\)Parameters:src - Source image.dst - Destination image of the same size and the same number of channels as src .ddepth - Desired depth of the destination image.ksize - Aperture size used to compute the second-derivative filters. See #getDerivKernels for details. The size must be positive and odd. applied. See #getDerivKernels for details. SEE: Sobel, Scharr

#### Laplacian public static void Laplacian([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int ksize, double scale) Calculates the Laplacian of an image. The function calculates the Laplacian of the source image by adding up the second x and y derivatives calculated using the Sobel operator: \(\texttt{dst} = \Delta \texttt{src} = \frac{\partial^2 \texttt{src}}{\partial x^2} + \frac{\partial^2 \texttt{src}}{\partial y^2}\) This is done when ksize > 1. When ksize == 1, the Laplacian is computed by filtering the image with the following \(3 \times 3\) aperture: \(\vecthreethree {0}{1}{0}{1}{-4}{1}{0}{1}{0}\)Parameters:src - Source image.dst - Destination image of the same size and the same number of channels as src .ddepth - Desired depth of the destination image.ksize - Aperture size used to compute the second-derivative filters. See #getDerivKernels for details. The size must be positive and odd.scale - Optional scale factor for the computed Laplacian values. By default, no scaling is applied. See #getDerivKernels for details. SEE: Sobel, Scharr

#### Laplacian public static void Laplacian([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int ksize, double scale, double delta) Calculates the Laplacian of an image. The function calculates the Laplacian of the source image by adding up the second x and y derivatives calculated using the Sobel operator: \(\texttt{dst} = \Delta \texttt{src} = \frac{\partial^2 \texttt{src}}{\partial x^2} + \frac{\partial^2 \texttt{src}}{\partial y^2}\) This is done when ksize > 1. When ksize == 1, the Laplacian is computed by filtering the image with the following \(3 \times 3\) aperture: \(\vecthreethree {0}{1}{0}{1}{-4}{1}{0}{1}{0}\)Parameters:src - Source image.dst - Destination image of the same size and the same number of channels as src .ddepth - Desired depth of the destination image.ksize - Aperture size used to compute the second-derivative filters. See #getDerivKernels for details. The size must be positive and odd.scale - Optional scale factor for the computed Laplacian values. By default, no scaling is applied. See #getDerivKernels for details.delta - Optional delta value that is added to the results prior to storing them in dst . SEE: Sobel, Scharr

#### Laplacian public static void Laplacian([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int ksize, double scale, double delta, int borderType) Calculates the Laplacian of an image. The function calculates the Laplacian of the source image by adding up the second x and y derivatives calculated using the Sobel operator: \(\texttt{dst} = \Delta \texttt{src} = \frac{\partial^2 \texttt{src}}{\partial x^2} + \frac{\partial^2 \texttt{src}}{\partial y^2}\) This is done when ksize > 1. When ksize == 1, the Laplacian is computed by filtering the image with the following \(3 \times 3\) aperture: \(\vecthreethree {0}{1}{0}{1}{-4}{1}{0}{1}{0}\)Parameters:src - Source image.dst - Destination image of the same size and the same number of channels as src .ddepth - Desired depth of the destination image.ksize - Aperture size used to compute the second-derivative filters. See #getDerivKernels for details. The size must be positive and odd.scale - Optional scale factor for the computed Laplacian values. By default, no scaling is applied. See #getDerivKernels for details.delta - Optional delta value that is added to the results prior to storing them in dst .borderType - Pixel extrapolation method, see #BorderTypes. #BORDER\_WRAP is not supported. SEE: Sobel, Scharr

#### line public static void line([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Draws a line segment connecting two points. The function line draws the line segment between pt1 and pt2 points in the image. The line is clipped by the image boundaries. For non-antialiased lines with integer coordinates, the 8-connected or 4-connected Bresenham algorithm is used. Thick lines are drawn with rounding endings. Antialiased lines are drawn using Gaussian filtering.Parameters:img - Image.pt1 - First point of the line segment.pt2 - Second point of the line segment.color - Line color.

#### line public static void line([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness) Draws a line segment connecting two points. The function line draws the line segment between pt1 and pt2 points in the image. The line is clipped by the image boundaries. For non-antialiased lines with integer coordinates, the 8-connected or 4-connected Bresenham algorithm is used. Thick lines are drawn with rounding endings. Antialiased lines are drawn using Gaussian filtering.Parameters:img - Image.pt1 - First point of the line segment.pt2 - Second point of the line segment.color - Line color.thickness - Line thickness.

#### line public static void line([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType) Draws a line segment connecting two points. The function line draws the line segment between pt1 and pt2 points in the image. The line is clipped by the image boundaries. For non-antialiased lines with integer coordinates, the 8-connected or 4-connected Bresenham algorithm is used. Thick lines are drawn with rounding endings. Antialiased lines are drawn using Gaussian filtering.Parameters:img - Image.pt1 - First point of the line segment.pt2 - Second point of the line segment.color - Line color.thickness - Line thickness.lineType - Type of the line. See #LineTypes.

#### line public static void line([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType, int shift) Draws a line segment connecting two points. The function line draws the line segment between pt1 and pt2 points in the image. The line is clipped by the image boundaries. For non-antialiased lines with integer coordinates, the 8-connected or 4-connected Bresenham algorithm is used. Thick lines are drawn with rounding endings. Antialiased lines are drawn using Gaussian filtering.Parameters:img - Image.pt1 - First point of the line segment.pt2 - Second point of the line segment.color - Line color.thickness - Line thickness.lineType - Type of the line. See #LineTypes.shift - Number of fractional bits in the point coordinates.

#### matchShapes public static double matchShapes([Mat](http://docs.google.com/org/opencv/core/Mat.html) contour1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) contour2, int method, double parameter) Compares two shapes. The function compares two shapes. All three implemented methods use the Hu invariants (see #HuMoments)Parameters:contour1 - First contour or grayscale image.contour2 - Second contour or grayscale image.method - Comparison method, see #ShapeMatchModesparameter - Method-specific parameter (not supported now). Returns:automatically generated

#### matchTemplate public static void matchTemplate([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) templ, [Mat](http://docs.google.com/org/opencv/core/Mat.html) result, int method) Compares a template against overlapped image regions. The function slides through image , compares the overlapped patches of size \(w \times h\) against templ using the specified method and stores the comparison results in result . Here are the formulae for the available comparison methods ( \(I\) denotes image, \(T\) template, \(R\) result ). The summation is done over template and/or the image patch: \(x' = 0...w-1, y' = 0...h-1\) After the function finishes the comparison, the best matches can be found as global minimums (when #TM\_SQDIFF was used) or maximums (when #TM\_CCORR or #TM\_CCOEFF was used) using the #minMaxLoc function. In case of a color image, template summation in the numerator and each sum in the denominator is done over all of the channels and separate mean values are used for each channel. That is, the function can take a color template and a color image. The result will still be a single-channel image, which is easier to analyze.Parameters:image - Image where the search is running. It must be 8-bit or 32-bit floating-point.templ - Searched template. It must be not greater than the source image and have the same data type.result - Map of comparison results. It must be single-channel 32-bit floating-point. If image is \(W \times H\) and templ is \(w \times h\) , then result is \((W-w+1) \times (H-h+1)\) .method - Parameter specifying the comparison method, see #TemplateMatchModes not set by default. Currently, only the #TM\_SQDIFF and #TM\_CCORR\_NORMED methods are supported.

#### matchTemplate public static void matchTemplate([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) templ, [Mat](http://docs.google.com/org/opencv/core/Mat.html) result, int method, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask) Compares a template against overlapped image regions. The function slides through image , compares the overlapped patches of size \(w \times h\) against templ using the specified method and stores the comparison results in result . Here are the formulae for the available comparison methods ( \(I\) denotes image, \(T\) template, \(R\) result ). The summation is done over template and/or the image patch: \(x' = 0...w-1, y' = 0...h-1\) After the function finishes the comparison, the best matches can be found as global minimums (when #TM\_SQDIFF was used) or maximums (when #TM\_CCORR or #TM\_CCOEFF was used) using the #minMaxLoc function. In case of a color image, template summation in the numerator and each sum in the denominator is done over all of the channels and separate mean values are used for each channel. That is, the function can take a color template and a color image. The result will still be a single-channel image, which is easier to analyze.Parameters:image - Image where the search is running. It must be 8-bit or 32-bit floating-point.templ - Searched template. It must be not greater than the source image and have the same data type.result - Map of comparison results. It must be single-channel 32-bit floating-point. If image is \(W \times H\) and templ is \(w \times h\) , then result is \((W-w+1) \times (H-h+1)\) .method - Parameter specifying the comparison method, see #TemplateMatchModesmask - Mask of searched template. It must have the same datatype and size with templ. It is not set by default. Currently, only the #TM\_SQDIFF and #TM\_CCORR\_NORMED methods are supported.

#### medianBlur public static void medianBlur([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ksize) Blurs an image using the median filter. The function smoothes an image using the median filter with the \(\texttt{ksize} \times \texttt{ksize}\) aperture. Each channel of a multi-channel image is processed independently. In-place operation is supported. **Note:** The median filter uses #BORDER\_REPLICATE internally to cope with border pixels, see #BorderTypesParameters:src - input 1-, 3-, or 4-channel image; when ksize is 3 or 5, the image depth should be CV\_8U, CV\_16U, or CV\_32F, for larger aperture sizes, it can only be CV\_8U.dst - destination array of the same size and type as src.ksize - aperture linear size; it must be odd and greater than 1, for example: 3, 5, 7 ... SEE: bilateralFilter, blur, boxFilter, GaussianBlur

#### minAreaRect public static [RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) minAreaRect([MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) points) Finds a rotated rectangle of the minimum area enclosing the input 2D point set. The function calculates and returns the minimum-area bounding rectangle (possibly rotated) for a specified point set. Developer should keep in mind that the returned RotatedRect can contain negative indices when data is close to the containing Mat element boundary.Parameters:points - Input vector of 2D points, stored in std::vector<> or Mat Returns:automatically generated

#### minEnclosingCircle public static void minEnclosingCircle([MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) points, [Point](http://docs.google.com/org/opencv/core/Point.html) center, float[] radius) Finds a circle of the minimum area enclosing a 2D point set. The function finds the minimal enclosing circle of a 2D point set using an iterative algorithm.Parameters:points - Input vector of 2D points, stored in std::vector<> or Matcenter - Output center of the circle.radius - Output radius of the circle.

#### minEnclosingTriangle public static double minEnclosingTriangle([Mat](http://docs.google.com/org/opencv/core/Mat.html) points, [Mat](http://docs.google.com/org/opencv/core/Mat.html) triangle) Finds a triangle of minimum area enclosing a 2D point set and returns its area. The function finds a triangle of minimum area enclosing the given set of 2D points and returns its area. The output for a given 2D point set is shown in the image below. 2D points are depicted in red\* and the enclosing triangle in \*yellow\*. ![Sample output of the minimum enclosing triangle function](pics/minenclosingtriangle.png) The implementation of the algorithm is based on O'Rourke's CITE: ORourke86 and Klee and Laskowski's CITE: KleeLaskowski85 papers. O'Rourke provides a \(\theta(n)\) algorithm for finding the minimal enclosing triangle of a 2D convex polygon with n vertices. Since the #minEnclosingTriangle function takes a 2D point set as input an additional preprocessing step of computing the convex hull of the 2D point set is required. The complexity of the #convexHull function is \(O(n log(n))\) which is higher than \(\theta(n)\). Thus the overall complexity of the function is \(O(n log(n))\).Parameters:points - Input vector of 2D points with depth CV\_32S or CV\_32F, stored in std::vector<> or Mattriangle - Output vector of three 2D points defining the vertices of the triangle. The depth of the OutputArray must be CV\_32F. Returns:automatically generated

#### moments public static [Moments](http://docs.google.com/org/opencv/imgproc/Moments.html) moments([Mat](http://docs.google.com/org/opencv/core/Mat.html) array) Calculates all of the moments up to the third order of a polygon or rasterized shape. The function computes moments, up to the 3rd order, of a vector shape or a rasterized shape. The results are returned in the structure cv::Moments.Parameters:array - Raster image (single-channel, 8-bit or floating-point 2D array) or an array ( \(1 \times N\) or \(N \times 1\) ) of 2D points (Point or Point2f ). used for images only. Returns:moments. **Note:** Only applicable to contour moments calculations from Python bindings: Note that the numpy type for the input array should be either np.int32 or np.float32. SEE: contourArea, arcLength

#### moments public static [Moments](http://docs.google.com/org/opencv/imgproc/Moments.html) moments([Mat](http://docs.google.com/org/opencv/core/Mat.html) array, boolean binaryImage) Calculates all of the moments up to the third order of a polygon or rasterized shape. The function computes moments, up to the 3rd order, of a vector shape or a rasterized shape. The results are returned in the structure cv::Moments.Parameters:array - Raster image (single-channel, 8-bit or floating-point 2D array) or an array ( \(1 \times N\) or \(N \times 1\) ) of 2D points (Point or Point2f ).binaryImage - If it is true, all non-zero image pixels are treated as 1's. The parameter is used for images only. Returns:moments. **Note:** Only applicable to contour moments calculations from Python bindings: Note that the numpy type for the input array should be either np.int32 or np.float32. SEE: contourArea, arcLength

#### morphologyEx public static void morphologyEx([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int op, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel) Performs advanced morphological transformations. The function cv::morphologyEx can perform advanced morphological transformations using an erosion and dilation as basic operations. Any of the operations can be done in-place. In case of multi-channel images, each channel is processed independently.Parameters:src - Source image. The number of channels can be arbitrary. The depth should be one of CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - Destination image of the same size and type as source image.op - Type of a morphological operation, see #MorphTypeskernel - Structuring element. It can be created using #getStructuringElement. kernel center. meaning. SEE: dilate, erode, getStructuringElement **Note:** The number of iterations is the number of times erosion or dilatation operation will be applied. For instance, an opening operation (#MORPH\_OPEN) with two iterations is equivalent to apply successively: erode -> erode -> dilate -> dilate (and not erode -> dilate -> erode -> dilate).

#### morphologyEx public static void morphologyEx([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int op, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor) Performs advanced morphological transformations. The function cv::morphologyEx can perform advanced morphological transformations using an erosion and dilation as basic operations. Any of the operations can be done in-place. In case of multi-channel images, each channel is processed independently.Parameters:src - Source image. The number of channels can be arbitrary. The depth should be one of CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - Destination image of the same size and type as source image.op - Type of a morphological operation, see #MorphTypeskernel - Structuring element. It can be created using #getStructuringElement.anchor - Anchor position with the kernel. Negative values mean that the anchor is at the kernel center. meaning. SEE: dilate, erode, getStructuringElement **Note:** The number of iterations is the number of times erosion or dilatation operation will be applied. For instance, an opening operation (#MORPH\_OPEN) with two iterations is equivalent to apply successively: erode -> erode -> dilate -> dilate (and not erode -> dilate -> erode -> dilate).

#### morphologyEx public static void morphologyEx([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int op, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int iterations) Performs advanced morphological transformations. The function cv::morphologyEx can perform advanced morphological transformations using an erosion and dilation as basic operations. Any of the operations can be done in-place. In case of multi-channel images, each channel is processed independently.Parameters:src - Source image. The number of channels can be arbitrary. The depth should be one of CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - Destination image of the same size and type as source image.op - Type of a morphological operation, see #MorphTypeskernel - Structuring element. It can be created using #getStructuringElement.anchor - Anchor position with the kernel. Negative values mean that the anchor is at the kernel center.iterations - Number of times erosion and dilation are applied. meaning. SEE: dilate, erode, getStructuringElement **Note:** The number of iterations is the number of times erosion or dilatation operation will be applied. For instance, an opening operation (#MORPH\_OPEN) with two iterations is equivalent to apply successively: erode -> erode -> dilate -> dilate (and not erode -> dilate -> erode -> dilate).

#### morphologyEx public static void morphologyEx([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int op, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int iterations, int borderType) Performs advanced morphological transformations. The function cv::morphologyEx can perform advanced morphological transformations using an erosion and dilation as basic operations. Any of the operations can be done in-place. In case of multi-channel images, each channel is processed independently.Parameters:src - Source image. The number of channels can be arbitrary. The depth should be one of CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - Destination image of the same size and type as source image.op - Type of a morphological operation, see #MorphTypeskernel - Structuring element. It can be created using #getStructuringElement.anchor - Anchor position with the kernel. Negative values mean that the anchor is at the kernel center.iterations - Number of times erosion and dilation are applied.borderType - Pixel extrapolation method, see #BorderTypes. #BORDER\_WRAP is not supported. meaning. SEE: dilate, erode, getStructuringElement **Note:** The number of iterations is the number of times erosion or dilatation operation will be applied. For instance, an opening operation (#MORPH\_OPEN) with two iterations is equivalent to apply successively: erode -> erode -> dilate -> dilate (and not erode -> dilate -> erode -> dilate).

#### morphologyEx public static void morphologyEx([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int op, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernel, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, int iterations, int borderType, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) borderValue) Performs advanced morphological transformations. The function cv::morphologyEx can perform advanced morphological transformations using an erosion and dilation as basic operations. Any of the operations can be done in-place. In case of multi-channel images, each channel is processed independently.Parameters:src - Source image. The number of channels can be arbitrary. The depth should be one of CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.dst - Destination image of the same size and type as source image.op - Type of a morphological operation, see #MorphTypeskernel - Structuring element. It can be created using #getStructuringElement.anchor - Anchor position with the kernel. Negative values mean that the anchor is at the kernel center.iterations - Number of times erosion and dilation are applied.borderType - Pixel extrapolation method, see #BorderTypes. #BORDER\_WRAP is not supported.borderValue - Border value in case of a constant border. The default value has a special meaning. SEE: dilate, erode, getStructuringElement **Note:** The number of iterations is the number of times erosion or dilatation operation will be applied. For instance, an opening operation (#MORPH\_OPEN) with two iterations is equivalent to apply successively: erode -> erode -> dilate -> dilate (and not erode -> dilate -> erode -> dilate).

#### phaseCorrelate public static [Point](http://docs.google.com/org/opencv/core/Point.html) phaseCorrelate([Mat](http://docs.google.com/org/opencv/core/Mat.html) src1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) src2)

The function is used to detect translational shifts that occur between two images. The operation takes advantage of the Fourier shift theorem for detecting the translational shift in the frequency domain. It can be used for fast image registration as well as motion estimation. For more information please see <http://en.wikipedia.org/wiki/Phase\_correlation> Calculates the cross-power spectrum of two supplied source arrays. The arrays are padded if needed with getOptimalDFTSize. The function performs the following equations:

* + - * First it applies a Hanning window (see <http://en.wikipedia.org/wiki/Hann\_function>) to each image to remove possible edge effects. This window is cached until the array size changes to speed up processing time.
      * Next it computes the forward DFTs of each source array: \(\mathbf{G}\_a = \mathcal{F}\{src\_1\}, \; \mathbf{G}\_b = \mathcal{F}\{src\_2\}\) where \(\mathcal{F}\) is the forward DFT.
      * It then computes the cross-power spectrum of each frequency domain array: \(R = \frac{ \mathbf{G}\_a \mathbf{G}\_b^\*}{|\mathbf{G}\_a \mathbf{G}\_b^\*|}\)
      * Next the cross-correlation is converted back into the time domain via the inverse DFT: \(r = \mathcal{F}^{-1}\{R\}\)
      * Finally, it computes the peak location and computes a 5x5 weighted centroid around the peak to achieve sub-pixel accuracy. \((\Delta x, \Delta y) = \texttt{weightedCentroid} \{\arg \max\_{(x, y)}\{r\}\}\)
      * If non-zero, the response parameter is computed as the sum of the elements of r within the 5x5 centroid around the peak location. It is normalized to a maximum of 1 (meaning there is a single peak) and will be smaller when there are multiple peaks.

Parameters:src1 - Source floating point array (CV\_32FC1 or CV\_64FC1)src2 - Source floating point array (CV\_32FC1 or CV\_64FC1) Returns:detected phase shift (sub-pixel) between the two arrays. SEE: dft, getOptimalDFTSize, idft, mulSpectrums createHanningWindow

#### phaseCorrelate public static [Point](http://docs.google.com/org/opencv/core/Point.html) phaseCorrelate([Mat](http://docs.google.com/org/opencv/core/Mat.html) src1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) src2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) window)

The function is used to detect translational shifts that occur between two images. The operation takes advantage of the Fourier shift theorem for detecting the translational shift in the frequency domain. It can be used for fast image registration as well as motion estimation. For more information please see <http://en.wikipedia.org/wiki/Phase\_correlation> Calculates the cross-power spectrum of two supplied source arrays. The arrays are padded if needed with getOptimalDFTSize. The function performs the following equations:

* + - * First it applies a Hanning window (see <http://en.wikipedia.org/wiki/Hann\_function>) to each image to remove possible edge effects. This window is cached until the array size changes to speed up processing time.
      * Next it computes the forward DFTs of each source array: \(\mathbf{G}\_a = \mathcal{F}\{src\_1\}, \; \mathbf{G}\_b = \mathcal{F}\{src\_2\}\) where \(\mathcal{F}\) is the forward DFT.
      * It then computes the cross-power spectrum of each frequency domain array: \(R = \frac{ \mathbf{G}\_a \mathbf{G}\_b^\*}{|\mathbf{G}\_a \mathbf{G}\_b^\*|}\)
      * Next the cross-correlation is converted back into the time domain via the inverse DFT: \(r = \mathcal{F}^{-1}\{R\}\)
      * Finally, it computes the peak location and computes a 5x5 weighted centroid around the peak to achieve sub-pixel accuracy. \((\Delta x, \Delta y) = \texttt{weightedCentroid} \{\arg \max\_{(x, y)}\{r\}\}\)
      * If non-zero, the response parameter is computed as the sum of the elements of r within the 5x5 centroid around the peak location. It is normalized to a maximum of 1 (meaning there is a single peak) and will be smaller when there are multiple peaks.

Parameters:src1 - Source floating point array (CV\_32FC1 or CV\_64FC1)src2 - Source floating point array (CV\_32FC1 or CV\_64FC1)window - Floating point array with windowing coefficients to reduce edge effects (optional). Returns:detected phase shift (sub-pixel) between the two arrays. SEE: dft, getOptimalDFTSize, idft, mulSpectrums createHanningWindow

#### phaseCorrelate public static [Point](http://docs.google.com/org/opencv/core/Point.html) phaseCorrelate([Mat](http://docs.google.com/org/opencv/core/Mat.html) src1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) src2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) window, double[] response)

The function is used to detect translational shifts that occur between two images. The operation takes advantage of the Fourier shift theorem for detecting the translational shift in the frequency domain. It can be used for fast image registration as well as motion estimation. For more information please see <http://en.wikipedia.org/wiki/Phase\_correlation> Calculates the cross-power spectrum of two supplied source arrays. The arrays are padded if needed with getOptimalDFTSize. The function performs the following equations:

* + - * First it applies a Hanning window (see <http://en.wikipedia.org/wiki/Hann\_function>) to each image to remove possible edge effects. This window is cached until the array size changes to speed up processing time.
      * Next it computes the forward DFTs of each source array: \(\mathbf{G}\_a = \mathcal{F}\{src\_1\}, \; \mathbf{G}\_b = \mathcal{F}\{src\_2\}\) where \(\mathcal{F}\) is the forward DFT.
      * It then computes the cross-power spectrum of each frequency domain array: \(R = \frac{ \mathbf{G}\_a \mathbf{G}\_b^\*}{|\mathbf{G}\_a \mathbf{G}\_b^\*|}\)
      * Next the cross-correlation is converted back into the time domain via the inverse DFT: \(r = \mathcal{F}^{-1}\{R\}\)
      * Finally, it computes the peak location and computes a 5x5 weighted centroid around the peak to achieve sub-pixel accuracy. \((\Delta x, \Delta y) = \texttt{weightedCentroid} \{\arg \max\_{(x, y)}\{r\}\}\)
      * If non-zero, the response parameter is computed as the sum of the elements of r within the 5x5 centroid around the peak location. It is normalized to a maximum of 1 (meaning there is a single peak) and will be smaller when there are multiple peaks.

Parameters:src1 - Source floating point array (CV\_32FC1 or CV\_64FC1)src2 - Source floating point array (CV\_32FC1 or CV\_64FC1)window - Floating point array with windowing coefficients to reduce edge effects (optional).response - Signal power within the 5x5 centroid around the peak, between 0 and 1 (optional). Returns:detected phase shift (sub-pixel) between the two arrays. SEE: dft, getOptimalDFTSize, idft, mulSpectrums createHanningWindow

#### pointPolygonTest public static double pointPolygonTest([MatOfPoint2f](http://docs.google.com/org/opencv/core/MatOfPoint2f.html) contour, [Point](http://docs.google.com/org/opencv/core/Point.html) pt, boolean measureDist) Performs a point-in-contour test. The function determines whether the point is inside a contour, outside, or lies on an edge (or coincides with a vertex). It returns positive (inside), negative (outside), or zero (on an edge) value, correspondingly. When measureDist=false , the return value is +1, -1, and 0, respectively. Otherwise, the return value is a signed distance between the point and the nearest contour edge. See below a sample output of the function where each image pixel is tested against the contour: ![sample output](pics/pointpolygon.png)Parameters:contour - Input contour.pt - Point tested against the contour.measureDist - If true, the function estimates the signed distance from the point to the nearest contour edge. Otherwise, the function only checks if the point is inside a contour or not. Returns:automatically generated

#### polylines public static void polylines([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> pts, boolean isClosed, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Draws several polygonal curves.Parameters:img - Image.pts - Array of polygonal curves.isClosed - Flag indicating whether the drawn polylines are closed or not. If they are closed, the function draws a line from the last vertex of each curve to its first vertex.color - Polyline color. The function cv::polylines draws one or more polygonal curves.

#### polylines public static void polylines([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> pts, boolean isClosed, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness) Draws several polygonal curves.Parameters:img - Image.pts - Array of polygonal curves.isClosed - Flag indicating whether the drawn polylines are closed or not. If they are closed, the function draws a line from the last vertex of each curve to its first vertex.color - Polyline color.thickness - Thickness of the polyline edges. The function cv::polylines draws one or more polygonal curves.

#### polylines public static void polylines([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> pts, boolean isClosed, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType) Draws several polygonal curves.Parameters:img - Image.pts - Array of polygonal curves.isClosed - Flag indicating whether the drawn polylines are closed or not. If they are closed, the function draws a line from the last vertex of each curve to its first vertex.color - Polyline color.thickness - Thickness of the polyline edges.lineType - Type of the line segments. See #LineTypes The function cv::polylines draws one or more polygonal curves.

#### polylines public static void polylines([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.util.List<[MatOfPoint](http://docs.google.com/org/opencv/core/MatOfPoint.html)> pts, boolean isClosed, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType, int shift) Draws several polygonal curves.Parameters:img - Image.pts - Array of polygonal curves.isClosed - Flag indicating whether the drawn polylines are closed or not. If they are closed, the function draws a line from the last vertex of each curve to its first vertex.color - Polyline color.thickness - Thickness of the polyline edges.lineType - Type of the line segments. See #LineTypesshift - Number of fractional bits in the vertex coordinates. The function cv::polylines draws one or more polygonal curves.

#### preCornerDetect public static void preCornerDetect([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ksize) Calculates a feature map for corner detection. The function calculates the complex spatial derivative-based function of the source image \(\texttt{dst} = (D\_x \texttt{src} )^2 \cdot D\_{yy} \texttt{src} + (D\_y \texttt{src} )^2 \cdot D\_{xx} \texttt{src} - 2 D\_x \texttt{src} \cdot D\_y \texttt{src} \cdot D\_{xy} \texttt{src}\) where \(D\_x\),\(D\_y\) are the first image derivatives, \(D\_{xx}\),\(D\_{yy}\) are the second image derivatives, and \(D\_{xy}\) is the mixed derivative. The corners can be found as local maximums of the functions, as shown below: Mat corners, dilated\_corners; preCornerDetect(image, corners, 3); // dilation with 3x3 rectangular structuring element dilate(corners, dilated\_corners, Mat(), 1); Mat corner\_mask = corners == dilated\_corners;Parameters:src - Source single-channel 8-bit of floating-point image.dst - Output image that has the type CV\_32F and the same size as src .ksize - %Aperture size of the Sobel .

#### preCornerDetect public static void preCornerDetect([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ksize, int borderType) Calculates a feature map for corner detection. The function calculates the complex spatial derivative-based function of the source image \(\texttt{dst} = (D\_x \texttt{src} )^2 \cdot D\_{yy} \texttt{src} + (D\_y \texttt{src} )^2 \cdot D\_{xx} \texttt{src} - 2 D\_x \texttt{src} \cdot D\_y \texttt{src} \cdot D\_{xy} \texttt{src}\) where \(D\_x\),\(D\_y\) are the first image derivatives, \(D\_{xx}\),\(D\_{yy}\) are the second image derivatives, and \(D\_{xy}\) is the mixed derivative. The corners can be found as local maximums of the functions, as shown below: Mat corners, dilated\_corners; preCornerDetect(image, corners, 3); // dilation with 3x3 rectangular structuring element dilate(corners, dilated\_corners, Mat(), 1); Mat corner\_mask = corners == dilated\_corners;Parameters:src - Source single-channel 8-bit of floating-point image.dst - Output image that has the type CV\_32F and the same size as src .ksize - %Aperture size of the Sobel .borderType - Pixel extrapolation method. See #BorderTypes. #BORDER\_WRAP is not supported.

#### putText public static void putText([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.lang.String text, [Point](http://docs.google.com/org/opencv/core/Point.html) org, int fontFace, double fontScale, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Draws a text string. The function cv::putText renders the specified text string in the image. Symbols that cannot be rendered using the specified font are replaced by question marks. See #getTextSize for a text rendering code example.Parameters:img - Image.text - Text string to be drawn.org - Bottom-left corner of the text string in the image.fontFace - Font type, see #HersheyFonts.fontScale - Font scale factor that is multiplied by the font-specific base size.color - Text color. it is at the top-left corner.

#### putText public static void putText([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.lang.String text, [Point](http://docs.google.com/org/opencv/core/Point.html) org, int fontFace, double fontScale, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness) Draws a text string. The function cv::putText renders the specified text string in the image. Symbols that cannot be rendered using the specified font are replaced by question marks. See #getTextSize for a text rendering code example.Parameters:img - Image.text - Text string to be drawn.org - Bottom-left corner of the text string in the image.fontFace - Font type, see #HersheyFonts.fontScale - Font scale factor that is multiplied by the font-specific base size.color - Text color.thickness - Thickness of the lines used to draw a text. it is at the top-left corner.

#### putText public static void putText([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.lang.String text, [Point](http://docs.google.com/org/opencv/core/Point.html) org, int fontFace, double fontScale, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType) Draws a text string. The function cv::putText renders the specified text string in the image. Symbols that cannot be rendered using the specified font are replaced by question marks. See #getTextSize for a text rendering code example.Parameters:img - Image.text - Text string to be drawn.org - Bottom-left corner of the text string in the image.fontFace - Font type, see #HersheyFonts.fontScale - Font scale factor that is multiplied by the font-specific base size.color - Text color.thickness - Thickness of the lines used to draw a text.lineType - Line type. See #LineTypes it is at the top-left corner.

#### putText public static void putText([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, java.lang.String text, [Point](http://docs.google.com/org/opencv/core/Point.html) org, int fontFace, double fontScale, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType, boolean bottomLeftOrigin) Draws a text string. The function cv::putText renders the specified text string in the image. Symbols that cannot be rendered using the specified font are replaced by question marks. See #getTextSize for a text rendering code example.Parameters:img - Image.text - Text string to be drawn.org - Bottom-left corner of the text string in the image.fontFace - Font type, see #HersheyFonts.fontScale - Font scale factor that is multiplied by the font-specific base size.color - Text color.thickness - Thickness of the lines used to draw a text.lineType - Line type. See #LineTypesbottomLeftOrigin - When true, the image data origin is at the bottom-left corner. Otherwise, it is at the top-left corner.

#### pyrDown public static void pyrDown([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Blurs an image and downsamples it. By default, size of the output image is computed as Size((src.cols+1)/2, (src.rows+1)/2), but in any case, the following conditions should be satisfied: \(\begin{array}{l} | \texttt{dstsize.width} \*2-src.cols| \leq 2 \\ | \texttt{dstsize.height} \*2-src.rows| \leq 2 \end{array}\) The function performs the downsampling step of the Gaussian pyramid construction. First, it convolves the source image with the kernel: \(\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}\) Then, it downsamples the image by rejecting even rows and columns.Parameters:src - input image.dst - output image; it has the specified size and the same type as src.

#### pyrDown public static void pyrDown([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) dstsize) Blurs an image and downsamples it. By default, size of the output image is computed as Size((src.cols+1)/2, (src.rows+1)/2), but in any case, the following conditions should be satisfied: \(\begin{array}{l} | \texttt{dstsize.width} \*2-src.cols| \leq 2 \\ | \texttt{dstsize.height} \*2-src.rows| \leq 2 \end{array}\) The function performs the downsampling step of the Gaussian pyramid construction. First, it convolves the source image with the kernel: \(\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}\) Then, it downsamples the image by rejecting even rows and columns.Parameters:src - input image.dst - output image; it has the specified size and the same type as src.dstsize - size of the output image.

#### pyrDown public static void pyrDown([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) dstsize, int borderType) Blurs an image and downsamples it. By default, size of the output image is computed as Size((src.cols+1)/2, (src.rows+1)/2), but in any case, the following conditions should be satisfied: \(\begin{array}{l} | \texttt{dstsize.width} \*2-src.cols| \leq 2 \\ | \texttt{dstsize.height} \*2-src.rows| \leq 2 \end{array}\) The function performs the downsampling step of the Gaussian pyramid construction. First, it convolves the source image with the kernel: \(\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}\) Then, it downsamples the image by rejecting even rows and columns.Parameters:src - input image.dst - output image; it has the specified size and the same type as src.dstsize - size of the output image.borderType - Pixel extrapolation method, see #BorderTypes (#BORDER\_CONSTANT isn't supported)

#### pyrMeanShiftFiltering public static void pyrMeanShiftFiltering([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, double sp, double sr) Performs initial step of meanshift segmentation of an image. The function implements the filtering stage of meanshift segmentation, that is, the output of the function is the filtered "posterized" image with color gradients and fine-grain texture flattened. At every pixel (X,Y) of the input image (or down-sized input image, see below) the function executes meanshift iterations, that is, the pixel (X,Y) neighborhood in the joint space-color hyperspace is considered: \((x,y): X- \texttt{sp} \le x \le X+ \texttt{sp} , Y- \texttt{sp} \le y \le Y+ \texttt{sp} , ||(R,G,B)-(r,g,b)|| \le \texttt{sr}\) where (R,G,B) and (r,g,b) are the vectors of color components at (X,Y) and (x,y), respectively (though, the algorithm does not depend on the color space used, so any 3-component color space can be used instead). Over the neighborhood the average spatial value (X',Y') and average color vector (R',G',B') are found and they act as the neighborhood center on the next iteration: \((X,Y)~(X',Y'), (R,G,B)~(R',G',B').\) After the iterations over, the color components of the initial pixel (that is, the pixel from where the iterations started) are set to the final value (average color at the last iteration): \(I(X,Y) <- (R\*,G\*,B\*)\) When maxLevel > 0, the gaussian pyramid of maxLevel+1 levels is built, and the above procedure is run on the smallest layer first. After that, the results are propagated to the larger layer and the iterations are run again only on those pixels where the layer colors differ by more than sr from the lower-resolution layer of the pyramid. That makes boundaries of color regions sharper. Note that the results will be actually different from the ones obtained by running the meanshift procedure on the whole original image (i.e. when maxLevel==0).Parameters:src - The source 8-bit, 3-channel image.dst - The destination image of the same format and the same size as the source.sp - The spatial window radius.sr - The color window radius.

#### pyrMeanShiftFiltering public static void pyrMeanShiftFiltering([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, double sp, double sr, int maxLevel) Performs initial step of meanshift segmentation of an image. The function implements the filtering stage of meanshift segmentation, that is, the output of the function is the filtered "posterized" image with color gradients and fine-grain texture flattened. At every pixel (X,Y) of the input image (or down-sized input image, see below) the function executes meanshift iterations, that is, the pixel (X,Y) neighborhood in the joint space-color hyperspace is considered: \((x,y): X- \texttt{sp} \le x \le X+ \texttt{sp} , Y- \texttt{sp} \le y \le Y+ \texttt{sp} , ||(R,G,B)-(r,g,b)|| \le \texttt{sr}\) where (R,G,B) and (r,g,b) are the vectors of color components at (X,Y) and (x,y), respectively (though, the algorithm does not depend on the color space used, so any 3-component color space can be used instead). Over the neighborhood the average spatial value (X',Y') and average color vector (R',G',B') are found and they act as the neighborhood center on the next iteration: \((X,Y)~(X',Y'), (R,G,B)~(R',G',B').\) After the iterations over, the color components of the initial pixel (that is, the pixel from where the iterations started) are set to the final value (average color at the last iteration): \(I(X,Y) <- (R\*,G\*,B\*)\) When maxLevel > 0, the gaussian pyramid of maxLevel+1 levels is built, and the above procedure is run on the smallest layer first. After that, the results are propagated to the larger layer and the iterations are run again only on those pixels where the layer colors differ by more than sr from the lower-resolution layer of the pyramid. That makes boundaries of color regions sharper. Note that the results will be actually different from the ones obtained by running the meanshift procedure on the whole original image (i.e. when maxLevel==0).Parameters:src - The source 8-bit, 3-channel image.dst - The destination image of the same format and the same size as the source.sp - The spatial window radius.sr - The color window radius.maxLevel - Maximum level of the pyramid for the segmentation.

#### pyrMeanShiftFiltering public static void pyrMeanShiftFiltering([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, double sp, double sr, int maxLevel, [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) termcrit) Performs initial step of meanshift segmentation of an image. The function implements the filtering stage of meanshift segmentation, that is, the output of the function is the filtered "posterized" image with color gradients and fine-grain texture flattened. At every pixel (X,Y) of the input image (or down-sized input image, see below) the function executes meanshift iterations, that is, the pixel (X,Y) neighborhood in the joint space-color hyperspace is considered: \((x,y): X- \texttt{sp} \le x \le X+ \texttt{sp} , Y- \texttt{sp} \le y \le Y+ \texttt{sp} , ||(R,G,B)-(r,g,b)|| \le \texttt{sr}\) where (R,G,B) and (r,g,b) are the vectors of color components at (X,Y) and (x,y), respectively (though, the algorithm does not depend on the color space used, so any 3-component color space can be used instead). Over the neighborhood the average spatial value (X',Y') and average color vector (R',G',B') are found and they act as the neighborhood center on the next iteration: \((X,Y)~(X',Y'), (R,G,B)~(R',G',B').\) After the iterations over, the color components of the initial pixel (that is, the pixel from where the iterations started) are set to the final value (average color at the last iteration): \(I(X,Y) <- (R\*,G\*,B\*)\) When maxLevel > 0, the gaussian pyramid of maxLevel+1 levels is built, and the above procedure is run on the smallest layer first. After that, the results are propagated to the larger layer and the iterations are run again only on those pixels where the layer colors differ by more than sr from the lower-resolution layer of the pyramid. That makes boundaries of color regions sharper. Note that the results will be actually different from the ones obtained by running the meanshift procedure on the whole original image (i.e. when maxLevel==0).Parameters:src - The source 8-bit, 3-channel image.dst - The destination image of the same format and the same size as the source.sp - The spatial window radius.sr - The color window radius.maxLevel - Maximum level of the pyramid for the segmentation.termcrit - Termination criteria: when to stop meanshift iterations.

#### pyrUp public static void pyrUp([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Upsamples an image and then blurs it. By default, size of the output image is computed as Size(src.cols\\*2, (src.rows\\*2), but in any case, the following conditions should be satisfied: \(\begin{array}{l} | \texttt{dstsize.width} -src.cols\*2| \leq ( \texttt{dstsize.width} \mod 2) \\ | \texttt{dstsize.height} -src.rows\*2| \leq ( \texttt{dstsize.height} \mod 2) \end{array}\) The function performs the upsampling step of the Gaussian pyramid construction, though it can actually be used to construct the Laplacian pyramid. First, it upsamples the source image by injecting even zero rows and columns and then convolves the result with the same kernel as in pyrDown multiplied by 4.Parameters:src - input image.dst - output image. It has the specified size and the same type as src .

#### pyrUp public static void pyrUp([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) dstsize) Upsamples an image and then blurs it. By default, size of the output image is computed as Size(src.cols\\*2, (src.rows\\*2), but in any case, the following conditions should be satisfied: \(\begin{array}{l} | \texttt{dstsize.width} -src.cols\*2| \leq ( \texttt{dstsize.width} \mod 2) \\ | \texttt{dstsize.height} -src.rows\*2| \leq ( \texttt{dstsize.height} \mod 2) \end{array}\) The function performs the upsampling step of the Gaussian pyramid construction, though it can actually be used to construct the Laplacian pyramid. First, it upsamples the source image by injecting even zero rows and columns and then convolves the result with the same kernel as in pyrDown multiplied by 4.Parameters:src - input image.dst - output image. It has the specified size and the same type as src .dstsize - size of the output image.

#### pyrUp public static void pyrUp([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) dstsize, int borderType) Upsamples an image and then blurs it. By default, size of the output image is computed as Size(src.cols\\*2, (src.rows\\*2), but in any case, the following conditions should be satisfied: \(\begin{array}{l} | \texttt{dstsize.width} -src.cols\*2| \leq ( \texttt{dstsize.width} \mod 2) \\ | \texttt{dstsize.height} -src.rows\*2| \leq ( \texttt{dstsize.height} \mod 2) \end{array}\) The function performs the upsampling step of the Gaussian pyramid construction, though it can actually be used to construct the Laplacian pyramid. First, it upsamples the source image by injecting even zero rows and columns and then convolves the result with the same kernel as in pyrDown multiplied by 4.Parameters:src - input image.dst - output image. It has the specified size and the same type as src .dstsize - size of the output image.borderType - Pixel extrapolation method, see #BorderTypes (only #BORDER\_DEFAULT is supported)

#### rectangle public static void rectangle([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color) Draws a simple, thick, or filled up-right rectangle. The function cv::rectangle draws a rectangle outline or a filled rectangle whose two opposite corners are pt1 and pt2.Parameters:img - Image.pt1 - Vertex of the rectangle.pt2 - Vertex of the rectangle opposite to pt1 .color - Rectangle color or brightness (grayscale image). mean that the function has to draw a filled rectangle.

#### rectangle public static void rectangle([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness) Draws a simple, thick, or filled up-right rectangle. The function cv::rectangle draws a rectangle outline or a filled rectangle whose two opposite corners are pt1 and pt2.Parameters:img - Image.pt1 - Vertex of the rectangle.pt2 - Vertex of the rectangle opposite to pt1 .color - Rectangle color or brightness (grayscale image).thickness - Thickness of lines that make up the rectangle. Negative values, like #FILLED, mean that the function has to draw a filled rectangle.

#### rectangle public static void rectangle([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType) Draws a simple, thick, or filled up-right rectangle. The function cv::rectangle draws a rectangle outline or a filled rectangle whose two opposite corners are pt1 and pt2.Parameters:img - Image.pt1 - Vertex of the rectangle.pt2 - Vertex of the rectangle opposite to pt1 .color - Rectangle color or brightness (grayscale image).thickness - Thickness of lines that make up the rectangle. Negative values, like #FILLED, mean that the function has to draw a filled rectangle.lineType - Type of the line. See #LineTypes

#### rectangle public static void rectangle([Mat](http://docs.google.com/org/opencv/core/Mat.html) img, [Point](http://docs.google.com/org/opencv/core/Point.html) pt1, [Point](http://docs.google.com/org/opencv/core/Point.html) pt2, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) color, int thickness, int lineType, int shift) Draws a simple, thick, or filled up-right rectangle. The function cv::rectangle draws a rectangle outline or a filled rectangle whose two opposite corners are pt1 and pt2.Parameters:img - Image.pt1 - Vertex of the rectangle.pt2 - Vertex of the rectangle opposite to pt1 .color - Rectangle color or brightness (grayscale image).thickness - Thickness of lines that make up the rectangle. Negative values, like #FILLED, mean that the function has to draw a filled rectangle.lineType - Type of the line. See #LineTypesshift - Number of fractional bits in the point coordinates.

#### remap public static void remap([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map2, int interpolation) Applies a generic geometrical transformation to an image. The function remap transforms the source image using the specified map: \(\texttt{dst} (x,y) = \texttt{src} (map\_x(x,y),map\_y(x,y))\) where values of pixels with non-integer coordinates are computed using one of available interpolation methods. \(map\_x\) and \(map\_y\) can be encoded as separate floating-point maps in \(map\_1\) and \(map\_2\) respectively, or interleaved floating-point maps of \((x,y)\) in \(map\_1\), or fixed-point maps created by using convertMaps. The reason you might want to convert from floating to fixed-point representations of a map is that they can yield much faster (\~2x) remapping operations. In the converted case, \(map\_1\) contains pairs (cvFloor(x), cvFloor(y)) and \(map\_2\) contains indices in a table of interpolation coefficients. This function cannot operate in-place.Parameters:src - Source image.dst - Destination image. It has the same size as map1 and the same type as src .map1 - The first map of either (x,y) points or just x values having the type CV\_16SC2 , CV\_32FC1, or CV\_32FC2. See convertMaps for details on converting a floating point representation to fixed-point for speed.map2 - The second map of y values having the type CV\_16UC1, CV\_32FC1, or none (empty map if map1 is (x,y) points), respectively.interpolation - Interpolation method (see #InterpolationFlags). The methods #INTER\_AREA and #INTER\_LINEAR\_EXACT are not supported by this function. borderMode=#BORDER\_TRANSPARENT, it means that the pixels in the destination image that corresponds to the "outliers" in the source image are not modified by the function. **Note:** Due to current implementation limitations the size of an input and output images should be less than 32767x32767.

#### remap public static void remap([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map2, int interpolation, int borderMode) Applies a generic geometrical transformation to an image. The function remap transforms the source image using the specified map: \(\texttt{dst} (x,y) = \texttt{src} (map\_x(x,y),map\_y(x,y))\) where values of pixels with non-integer coordinates are computed using one of available interpolation methods. \(map\_x\) and \(map\_y\) can be encoded as separate floating-point maps in \(map\_1\) and \(map\_2\) respectively, or interleaved floating-point maps of \((x,y)\) in \(map\_1\), or fixed-point maps created by using convertMaps. The reason you might want to convert from floating to fixed-point representations of a map is that they can yield much faster (\~2x) remapping operations. In the converted case, \(map\_1\) contains pairs (cvFloor(x), cvFloor(y)) and \(map\_2\) contains indices in a table of interpolation coefficients. This function cannot operate in-place.Parameters:src - Source image.dst - Destination image. It has the same size as map1 and the same type as src .map1 - The first map of either (x,y) points or just x values having the type CV\_16SC2 , CV\_32FC1, or CV\_32FC2. See convertMaps for details on converting a floating point representation to fixed-point for speed.map2 - The second map of y values having the type CV\_16UC1, CV\_32FC1, or none (empty map if map1 is (x,y) points), respectively.interpolation - Interpolation method (see #InterpolationFlags). The methods #INTER\_AREA and #INTER\_LINEAR\_EXACT are not supported by this function.borderMode - Pixel extrapolation method (see #BorderTypes). When borderMode=#BORDER\_TRANSPARENT, it means that the pixels in the destination image that corresponds to the "outliers" in the source image are not modified by the function. **Note:** Due to current implementation limitations the size of an input and output images should be less than 32767x32767.

#### remap public static void remap([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) map2, int interpolation, int borderMode, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) borderValue) Applies a generic geometrical transformation to an image. The function remap transforms the source image using the specified map: \(\texttt{dst} (x,y) = \texttt{src} (map\_x(x,y),map\_y(x,y))\) where values of pixels with non-integer coordinates are computed using one of available interpolation methods. \(map\_x\) and \(map\_y\) can be encoded as separate floating-point maps in \(map\_1\) and \(map\_2\) respectively, or interleaved floating-point maps of \((x,y)\) in \(map\_1\), or fixed-point maps created by using convertMaps. The reason you might want to convert from floating to fixed-point representations of a map is that they can yield much faster (\~2x) remapping operations. In the converted case, \(map\_1\) contains pairs (cvFloor(x), cvFloor(y)) and \(map\_2\) contains indices in a table of interpolation coefficients. This function cannot operate in-place.Parameters:src - Source image.dst - Destination image. It has the same size as map1 and the same type as src .map1 - The first map of either (x,y) points or just x values having the type CV\_16SC2 , CV\_32FC1, or CV\_32FC2. See convertMaps for details on converting a floating point representation to fixed-point for speed.map2 - The second map of y values having the type CV\_16UC1, CV\_32FC1, or none (empty map if map1 is (x,y) points), respectively.interpolation - Interpolation method (see #InterpolationFlags). The methods #INTER\_AREA and #INTER\_LINEAR\_EXACT are not supported by this function.borderMode - Pixel extrapolation method (see #BorderTypes). When borderMode=#BORDER\_TRANSPARENT, it means that the pixels in the destination image that corresponds to the "outliers" in the source image are not modified by the function.borderValue - Value used in case of a constant border. By default, it is 0. **Note:** Due to current implementation limitations the size of an input and output images should be less than 32767x32767.

#### resize public static void resize([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize) Resizes an image. The function resize resizes the image src down to or up to the specified size. Note that the initial dst type or size are not taken into account. Instead, the size and type are derived from the src,dsize,fx, and fy. If you want to resize src so that it fits the pre-created dst, you may call the function as follows: // explicitly specify dsize=dst.size(); fx and fy will be computed from that. resize(src, dst, dst.size(), 0, 0, interpolation); If you want to decimate the image by factor of 2 in each direction, you can call the function this way: // specify fx and fy and let the function compute the destination image size. resize(src, dst, Size(), 0.5, 0.5, interpolation); To shrink an image, it will generally look best with #INTER\_AREA interpolation, whereas to enlarge an image, it will generally look best with c#INTER\_CUBIC (slow) or #INTER\_LINEAR (faster but still looks OK).Parameters:src - input image.dst - output image; it has the size dsize (when it is non-zero) or the size computed from src.size(), fx, and fy; the type of dst is the same as of src.dsize - output image size; if it equals zero, it is computed as: \(\texttt{dsize = Size(round(fx\*src.cols), round(fy\*src.rows))}\) Either dsize or both fx and fy must be non-zero. \(\texttt{(double)dsize.width/src.cols}\) \(\texttt{(double)dsize.height/src.rows}\) SEE: warpAffine, warpPerspective, remap

#### resize public static void resize([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, double fx) Resizes an image. The function resize resizes the image src down to or up to the specified size. Note that the initial dst type or size are not taken into account. Instead, the size and type are derived from the src,dsize,fx, and fy. If you want to resize src so that it fits the pre-created dst, you may call the function as follows: // explicitly specify dsize=dst.size(); fx and fy will be computed from that. resize(src, dst, dst.size(), 0, 0, interpolation); If you want to decimate the image by factor of 2 in each direction, you can call the function this way: // specify fx and fy and let the function compute the destination image size. resize(src, dst, Size(), 0.5, 0.5, interpolation); To shrink an image, it will generally look best with #INTER\_AREA interpolation, whereas to enlarge an image, it will generally look best with c#INTER\_CUBIC (slow) or #INTER\_LINEAR (faster but still looks OK).Parameters:src - input image.dst - output image; it has the size dsize (when it is non-zero) or the size computed from src.size(), fx, and fy; the type of dst is the same as of src.dsize - output image size; if it equals zero, it is computed as: \(\texttt{dsize = Size(round(fx\*src.cols), round(fy\*src.rows))}\) Either dsize or both fx and fy must be non-zero.fx - scale factor along the horizontal axis; when it equals 0, it is computed as \(\texttt{(double)dsize.width/src.cols}\) \(\texttt{(double)dsize.height/src.rows}\) SEE: warpAffine, warpPerspective, remap

#### resize public static void resize([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, double fx, double fy) Resizes an image. The function resize resizes the image src down to or up to the specified size. Note that the initial dst type or size are not taken into account. Instead, the size and type are derived from the src,dsize,fx, and fy. If you want to resize src so that it fits the pre-created dst, you may call the function as follows: // explicitly specify dsize=dst.size(); fx and fy will be computed from that. resize(src, dst, dst.size(), 0, 0, interpolation); If you want to decimate the image by factor of 2 in each direction, you can call the function this way: // specify fx and fy and let the function compute the destination image size. resize(src, dst, Size(), 0.5, 0.5, interpolation); To shrink an image, it will generally look best with #INTER\_AREA interpolation, whereas to enlarge an image, it will generally look best with c#INTER\_CUBIC (slow) or #INTER\_LINEAR (faster but still looks OK).Parameters:src - input image.dst - output image; it has the size dsize (when it is non-zero) or the size computed from src.size(), fx, and fy; the type of dst is the same as of src.dsize - output image size; if it equals zero, it is computed as: \(\texttt{dsize = Size(round(fx\*src.cols), round(fy\*src.rows))}\) Either dsize or both fx and fy must be non-zero.fx - scale factor along the horizontal axis; when it equals 0, it is computed as \(\texttt{(double)dsize.width/src.cols}\)fy - scale factor along the vertical axis; when it equals 0, it is computed as \(\texttt{(double)dsize.height/src.rows}\) SEE: warpAffine, warpPerspective, remap

#### resize public static void resize([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, double fx, double fy, int interpolation) Resizes an image. The function resize resizes the image src down to or up to the specified size. Note that the initial dst type or size are not taken into account. Instead, the size and type are derived from the src,dsize,fx, and fy. If you want to resize src so that it fits the pre-created dst, you may call the function as follows: // explicitly specify dsize=dst.size(); fx and fy will be computed from that. resize(src, dst, dst.size(), 0, 0, interpolation); If you want to decimate the image by factor of 2 in each direction, you can call the function this way: // specify fx and fy and let the function compute the destination image size. resize(src, dst, Size(), 0.5, 0.5, interpolation); To shrink an image, it will generally look best with #INTER\_AREA interpolation, whereas to enlarge an image, it will generally look best with c#INTER\_CUBIC (slow) or #INTER\_LINEAR (faster but still looks OK).Parameters:src - input image.dst - output image; it has the size dsize (when it is non-zero) or the size computed from src.size(), fx, and fy; the type of dst is the same as of src.dsize - output image size; if it equals zero, it is computed as: \(\texttt{dsize = Size(round(fx\*src.cols), round(fy\*src.rows))}\) Either dsize or both fx and fy must be non-zero.fx - scale factor along the horizontal axis; when it equals 0, it is computed as \(\texttt{(double)dsize.width/src.cols}\)fy - scale factor along the vertical axis; when it equals 0, it is computed as \(\texttt{(double)dsize.height/src.rows}\)interpolation - interpolation method, see #InterpolationFlags SEE: warpAffine, warpPerspective, remap

#### rotatedRectangleIntersection public static int rotatedRectangleIntersection([RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) rect1, [RotatedRect](http://docs.google.com/org/opencv/core/RotatedRect.html) rect2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) intersectingRegion) Finds out if there is any intersection between two rotated rectangles. If there is then the vertices of the intersecting region are returned as well. Below are some examples of intersection configurations. The hatched pattern indicates the intersecting region and the red vertices are returned by the function. ![intersection examples](pics/intersection.png)Parameters:rect1 - First rectanglerect2 - Second rectangleintersectingRegion - The output array of the vertices of the intersecting region. It returns at most 8 vertices. Stored as std::vector<cv::Point2f> or cv::Mat as Mx1 of type CV\_32FC2. Returns:One of #RectanglesIntersectTypes

#### Scharr public static void Scharr([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int dx, int dy) Calculates the first x- or y- image derivative using Scharr operator. The function computes the first x- or y- spatial image derivative using the Scharr operator. The call \(\texttt{Scharr(src, dst, ddepth, dx, dy, scale, delta, borderType)}\) is equivalent to \(\texttt{Sobel(src, dst, ddepth, dx, dy, CV\_SCHARR, scale, delta, borderType)} .\)Parameters:src - input image.dst - output image of the same size and the same number of channels as src.ddepth - output image depth, see REF: filter\_depths "combinations"dx - order of the derivative x.dy - order of the derivative y. applied (see #getDerivKernels for details). SEE: cartToPolar

#### Scharr public static void Scharr([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int dx, int dy, double scale) Calculates the first x- or y- image derivative using Scharr operator. The function computes the first x- or y- spatial image derivative using the Scharr operator. The call \(\texttt{Scharr(src, dst, ddepth, dx, dy, scale, delta, borderType)}\) is equivalent to \(\texttt{Sobel(src, dst, ddepth, dx, dy, CV\_SCHARR, scale, delta, borderType)} .\)Parameters:src - input image.dst - output image of the same size and the same number of channels as src.ddepth - output image depth, see REF: filter\_depths "combinations"dx - order of the derivative x.dy - order of the derivative y.scale - optional scale factor for the computed derivative values; by default, no scaling is applied (see #getDerivKernels for details). SEE: cartToPolar

#### Scharr public static void Scharr([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int dx, int dy, double scale, double delta) Calculates the first x- or y- image derivative using Scharr operator. The function computes the first x- or y- spatial image derivative using the Scharr operator. The call \(\texttt{Scharr(src, dst, ddepth, dx, dy, scale, delta, borderType)}\) is equivalent to \(\texttt{Sobel(src, dst, ddepth, dx, dy, CV\_SCHARR, scale, delta, borderType)} .\)Parameters:src - input image.dst - output image of the same size and the same number of channels as src.ddepth - output image depth, see REF: filter\_depths "combinations"dx - order of the derivative x.dy - order of the derivative y.scale - optional scale factor for the computed derivative values; by default, no scaling is applied (see #getDerivKernels for details).delta - optional delta value that is added to the results prior to storing them in dst. SEE: cartToPolar

#### Scharr public static void Scharr([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int dx, int dy, double scale, double delta, int borderType) Calculates the first x- or y- image derivative using Scharr operator. The function computes the first x- or y- spatial image derivative using the Scharr operator. The call \(\texttt{Scharr(src, dst, ddepth, dx, dy, scale, delta, borderType)}\) is equivalent to \(\texttt{Sobel(src, dst, ddepth, dx, dy, CV\_SCHARR, scale, delta, borderType)} .\)Parameters:src - input image.dst - output image of the same size and the same number of channels as src.ddepth - output image depth, see REF: filter\_depths "combinations"dx - order of the derivative x.dy - order of the derivative y.scale - optional scale factor for the computed derivative values; by default, no scaling is applied (see #getDerivKernels for details).delta - optional delta value that is added to the results prior to storing them in dst.borderType - pixel extrapolation method, see #BorderTypes. #BORDER\_WRAP is not supported. SEE: cartToPolar

#### sepFilter2D public static void sepFilter2D([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernelX, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernelY) Applies a separable linear filter to an image. The function applies a separable linear filter to the image. That is, first, every row of src is filtered with the 1D kernel kernelX. Then, every column of the result is filtered with the 1D kernel kernelY. The final result shifted by delta is stored in dst .Parameters:src - Source image.dst - Destination image of the same size and the same number of channels as src .ddepth - Destination image depth, see REF: filter\_depths "combinations"kernelX - Coefficients for filtering each row.kernelY - Coefficients for filtering each column. is at the kernel center. SEE: filter2D, Sobel, GaussianBlur, boxFilter, blur

#### sepFilter2D public static void sepFilter2D([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernelX, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernelY, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor) Applies a separable linear filter to an image. The function applies a separable linear filter to the image. That is, first, every row of src is filtered with the 1D kernel kernelX. Then, every column of the result is filtered with the 1D kernel kernelY. The final result shifted by delta is stored in dst .Parameters:src - Source image.dst - Destination image of the same size and the same number of channels as src .ddepth - Destination image depth, see REF: filter\_depths "combinations"kernelX - Coefficients for filtering each row.kernelY - Coefficients for filtering each column.anchor - Anchor position within the kernel. The default value \((-1,-1)\) means that the anchor is at the kernel center. SEE: filter2D, Sobel, GaussianBlur, boxFilter, blur

#### sepFilter2D public static void sepFilter2D([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernelX, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernelY, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, double delta) Applies a separable linear filter to an image. The function applies a separable linear filter to the image. That is, first, every row of src is filtered with the 1D kernel kernelX. Then, every column of the result is filtered with the 1D kernel kernelY. The final result shifted by delta is stored in dst .Parameters:src - Source image.dst - Destination image of the same size and the same number of channels as src .ddepth - Destination image depth, see REF: filter\_depths "combinations"kernelX - Coefficients for filtering each row.kernelY - Coefficients for filtering each column.anchor - Anchor position within the kernel. The default value \((-1,-1)\) means that the anchor is at the kernel center.delta - Value added to the filtered results before storing them. SEE: filter2D, Sobel, GaussianBlur, boxFilter, blur

#### sepFilter2D public static void sepFilter2D([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernelX, [Mat](http://docs.google.com/org/opencv/core/Mat.html) kernelY, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, double delta, int borderType) Applies a separable linear filter to an image. The function applies a separable linear filter to the image. That is, first, every row of src is filtered with the 1D kernel kernelX. Then, every column of the result is filtered with the 1D kernel kernelY. The final result shifted by delta is stored in dst .Parameters:src - Source image.dst - Destination image of the same size and the same number of channels as src .ddepth - Destination image depth, see REF: filter\_depths "combinations"kernelX - Coefficients for filtering each row.kernelY - Coefficients for filtering each column.anchor - Anchor position within the kernel. The default value \((-1,-1)\) means that the anchor is at the kernel center.delta - Value added to the filtered results before storing them.borderType - Pixel extrapolation method, see #BorderTypes. #BORDER\_WRAP is not supported. SEE: filter2D, Sobel, GaussianBlur, boxFilter, blur

#### Sobel public static void Sobel([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int dx, int dy) Calculates the first, second, third, or mixed image derivatives using an extended Sobel operator. In all cases except one, the \(\texttt{ksize} \times \texttt{ksize}\) separable kernel is used to calculate the derivative. When \(\texttt{ksize = 1}\), the \(3 \times 1\) or \(1 \times 3\) kernel is used (that is, no Gaussian smoothing is done). ksize = 1 can only be used for the first or the second x- or y- derivatives. There is also the special value ksize = #CV\_SCHARR (-1) that corresponds to the \(3\times3\) Scharr filter that may give more accurate results than the \(3\times3\) Sobel. The Scharr aperture is \(\vecthreethree{-3}{0}{3}{-10}{0}{10}{-3}{0}{3}\) for the x-derivative, or transposed for the y-derivative. The function calculates an image derivative by convolving the image with the appropriate kernel: \(\texttt{dst} = \frac{\partial^{xorder+yorder} \texttt{src}}{\partial x^{xorder} \partial y^{yorder}}\) The Sobel operators combine Gaussian smoothing and differentiation, so the result is more or less resistant to the noise. Most often, the function is called with ( xorder = 1, yorder = 0, ksize = 3) or ( xorder = 0, yorder = 1, ksize = 3) to calculate the first x- or y- image derivative. The first case corresponds to a kernel of: \(\vecthreethree{-1}{0}{1}{-2}{0}{2}{-1}{0}{1}\) The second case corresponds to a kernel of: \(\vecthreethree{-1}{-2}{-1}{0}{0}{0}{1}{2}{1}\)Parameters:src - input image.dst - output image of the same size and the same number of channels as src .ddepth - output image depth, see REF: filter\_depths "combinations"; in the case of 8-bit input images it will result in truncated derivatives.dx - order of the derivative x.dy - order of the derivative y. applied (see #getDerivKernels for details). SEE: Scharr, Laplacian, sepFilter2D, filter2D, GaussianBlur, cartToPolar

#### Sobel public static void Sobel([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int dx, int dy, int ksize) Calculates the first, second, third, or mixed image derivatives using an extended Sobel operator. In all cases except one, the \(\texttt{ksize} \times \texttt{ksize}\) separable kernel is used to calculate the derivative. When \(\texttt{ksize = 1}\), the \(3 \times 1\) or \(1 \times 3\) kernel is used (that is, no Gaussian smoothing is done). ksize = 1 can only be used for the first or the second x- or y- derivatives. There is also the special value ksize = #CV\_SCHARR (-1) that corresponds to the \(3\times3\) Scharr filter that may give more accurate results than the \(3\times3\) Sobel. The Scharr aperture is \(\vecthreethree{-3}{0}{3}{-10}{0}{10}{-3}{0}{3}\) for the x-derivative, or transposed for the y-derivative. The function calculates an image derivative by convolving the image with the appropriate kernel: \(\texttt{dst} = \frac{\partial^{xorder+yorder} \texttt{src}}{\partial x^{xorder} \partial y^{yorder}}\) The Sobel operators combine Gaussian smoothing and differentiation, so the result is more or less resistant to the noise. Most often, the function is called with ( xorder = 1, yorder = 0, ksize = 3) or ( xorder = 0, yorder = 1, ksize = 3) to calculate the first x- or y- image derivative. The first case corresponds to a kernel of: \(\vecthreethree{-1}{0}{1}{-2}{0}{2}{-1}{0}{1}\) The second case corresponds to a kernel of: \(\vecthreethree{-1}{-2}{-1}{0}{0}{0}{1}{2}{1}\)Parameters:src - input image.dst - output image of the same size and the same number of channels as src .ddepth - output image depth, see REF: filter\_depths "combinations"; in the case of 8-bit input images it will result in truncated derivatives.dx - order of the derivative x.dy - order of the derivative y.ksize - size of the extended Sobel kernel; it must be 1, 3, 5, or 7. applied (see #getDerivKernels for details). SEE: Scharr, Laplacian, sepFilter2D, filter2D, GaussianBlur, cartToPolar

#### Sobel public static void Sobel([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int dx, int dy, int ksize, double scale) Calculates the first, second, third, or mixed image derivatives using an extended Sobel operator. In all cases except one, the \(\texttt{ksize} \times \texttt{ksize}\) separable kernel is used to calculate the derivative. When \(\texttt{ksize = 1}\), the \(3 \times 1\) or \(1 \times 3\) kernel is used (that is, no Gaussian smoothing is done). ksize = 1 can only be used for the first or the second x- or y- derivatives. There is also the special value ksize = #CV\_SCHARR (-1) that corresponds to the \(3\times3\) Scharr filter that may give more accurate results than the \(3\times3\) Sobel. The Scharr aperture is \(\vecthreethree{-3}{0}{3}{-10}{0}{10}{-3}{0}{3}\) for the x-derivative, or transposed for the y-derivative. The function calculates an image derivative by convolving the image with the appropriate kernel: \(\texttt{dst} = \frac{\partial^{xorder+yorder} \texttt{src}}{\partial x^{xorder} \partial y^{yorder}}\) The Sobel operators combine Gaussian smoothing and differentiation, so the result is more or less resistant to the noise. Most often, the function is called with ( xorder = 1, yorder = 0, ksize = 3) or ( xorder = 0, yorder = 1, ksize = 3) to calculate the first x- or y- image derivative. The first case corresponds to a kernel of: \(\vecthreethree{-1}{0}{1}{-2}{0}{2}{-1}{0}{1}\) The second case corresponds to a kernel of: \(\vecthreethree{-1}{-2}{-1}{0}{0}{0}{1}{2}{1}\)Parameters:src - input image.dst - output image of the same size and the same number of channels as src .ddepth - output image depth, see REF: filter\_depths "combinations"; in the case of 8-bit input images it will result in truncated derivatives.dx - order of the derivative x.dy - order of the derivative y.ksize - size of the extended Sobel kernel; it must be 1, 3, 5, or 7.scale - optional scale factor for the computed derivative values; by default, no scaling is applied (see #getDerivKernels for details). SEE: Scharr, Laplacian, sepFilter2D, filter2D, GaussianBlur, cartToPolar

#### Sobel public static void Sobel([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int dx, int dy, int ksize, double scale, double delta) Calculates the first, second, third, or mixed image derivatives using an extended Sobel operator. In all cases except one, the \(\texttt{ksize} \times \texttt{ksize}\) separable kernel is used to calculate the derivative. When \(\texttt{ksize = 1}\), the \(3 \times 1\) or \(1 \times 3\) kernel is used (that is, no Gaussian smoothing is done). ksize = 1 can only be used for the first or the second x- or y- derivatives. There is also the special value ksize = #CV\_SCHARR (-1) that corresponds to the \(3\times3\) Scharr filter that may give more accurate results than the \(3\times3\) Sobel. The Scharr aperture is \(\vecthreethree{-3}{0}{3}{-10}{0}{10}{-3}{0}{3}\) for the x-derivative, or transposed for the y-derivative. The function calculates an image derivative by convolving the image with the appropriate kernel: \(\texttt{dst} = \frac{\partial^{xorder+yorder} \texttt{src}}{\partial x^{xorder} \partial y^{yorder}}\) The Sobel operators combine Gaussian smoothing and differentiation, so the result is more or less resistant to the noise. Most often, the function is called with ( xorder = 1, yorder = 0, ksize = 3) or ( xorder = 0, yorder = 1, ksize = 3) to calculate the first x- or y- image derivative. The first case corresponds to a kernel of: \(\vecthreethree{-1}{0}{1}{-2}{0}{2}{-1}{0}{1}\) The second case corresponds to a kernel of: \(\vecthreethree{-1}{-2}{-1}{0}{0}{0}{1}{2}{1}\)Parameters:src - input image.dst - output image of the same size and the same number of channels as src .ddepth - output image depth, see REF: filter\_depths "combinations"; in the case of 8-bit input images it will result in truncated derivatives.dx - order of the derivative x.dy - order of the derivative y.ksize - size of the extended Sobel kernel; it must be 1, 3, 5, or 7.scale - optional scale factor for the computed derivative values; by default, no scaling is applied (see #getDerivKernels for details).delta - optional delta value that is added to the results prior to storing them in dst. SEE: Scharr, Laplacian, sepFilter2D, filter2D, GaussianBlur, cartToPolar

#### Sobel public static void Sobel([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, int dx, int dy, int ksize, double scale, double delta, int borderType) Calculates the first, second, third, or mixed image derivatives using an extended Sobel operator. In all cases except one, the \(\texttt{ksize} \times \texttt{ksize}\) separable kernel is used to calculate the derivative. When \(\texttt{ksize = 1}\), the \(3 \times 1\) or \(1 \times 3\) kernel is used (that is, no Gaussian smoothing is done). ksize = 1 can only be used for the first or the second x- or y- derivatives. There is also the special value ksize = #CV\_SCHARR (-1) that corresponds to the \(3\times3\) Scharr filter that may give more accurate results than the \(3\times3\) Sobel. The Scharr aperture is \(\vecthreethree{-3}{0}{3}{-10}{0}{10}{-3}{0}{3}\) for the x-derivative, or transposed for the y-derivative. The function calculates an image derivative by convolving the image with the appropriate kernel: \(\texttt{dst} = \frac{\partial^{xorder+yorder} \texttt{src}}{\partial x^{xorder} \partial y^{yorder}}\) The Sobel operators combine Gaussian smoothing and differentiation, so the result is more or less resistant to the noise. Most often, the function is called with ( xorder = 1, yorder = 0, ksize = 3) or ( xorder = 0, yorder = 1, ksize = 3) to calculate the first x- or y- image derivative. The first case corresponds to a kernel of: \(\vecthreethree{-1}{0}{1}{-2}{0}{2}{-1}{0}{1}\) The second case corresponds to a kernel of: \(\vecthreethree{-1}{-2}{-1}{0}{0}{0}{1}{2}{1}\)Parameters:src - input image.dst - output image of the same size and the same number of channels as src .ddepth - output image depth, see REF: filter\_depths "combinations"; in the case of 8-bit input images it will result in truncated derivatives.dx - order of the derivative x.dy - order of the derivative y.ksize - size of the extended Sobel kernel; it must be 1, 3, 5, or 7.scale - optional scale factor for the computed derivative values; by default, no scaling is applied (see #getDerivKernels for details).delta - optional delta value that is added to the results prior to storing them in dst.borderType - pixel extrapolation method, see #BorderTypes. #BORDER\_WRAP is not supported. SEE: Scharr, Laplacian, sepFilter2D, filter2D, GaussianBlur, cartToPolar

#### spatialGradient public static void spatialGradient([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dx, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dy) Calculates the first order image derivative in both x and y using a Sobel operator Equivalent to calling: Sobel( src, dx, CV\_16SC1, 1, 0, 3 ); Sobel( src, dy, CV\_16SC1, 0, 1, 3 );Parameters:src - input image.dx - output image with first-order derivative in x.dy - output image with first-order derivative in y. Only #BORDER\_DEFAULT=#BORDER\_REFLECT\_101 and #BORDER\_REPLICATE are supported. SEE: Sobel

#### spatialGradient public static void spatialGradient([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dx, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dy, int ksize) Calculates the first order image derivative in both x and y using a Sobel operator Equivalent to calling: Sobel( src, dx, CV\_16SC1, 1, 0, 3 ); Sobel( src, dy, CV\_16SC1, 0, 1, 3 );Parameters:src - input image.dx - output image with first-order derivative in x.dy - output image with first-order derivative in y.ksize - size of Sobel kernel. It must be 3. Only #BORDER\_DEFAULT=#BORDER\_REFLECT\_101 and #BORDER\_REPLICATE are supported. SEE: Sobel

#### spatialGradient public static void spatialGradient([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dx, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dy, int ksize, int borderType) Calculates the first order image derivative in both x and y using a Sobel operator Equivalent to calling: Sobel( src, dx, CV\_16SC1, 1, 0, 3 ); Sobel( src, dy, CV\_16SC1, 0, 1, 3 );Parameters:src - input image.dx - output image with first-order derivative in x.dy - output image with first-order derivative in y.ksize - size of Sobel kernel. It must be 3.borderType - pixel extrapolation method, see #BorderTypes. Only #BORDER\_DEFAULT=#BORDER\_REFLECT\_101 and #BORDER\_REPLICATE are supported. SEE: Sobel

#### sqrBoxFilter public static void sqrBoxFilter([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize) Calculates the normalized sum of squares of the pixel values overlapping the filter. For every pixel \( (x, y) \) in the source image, the function calculates the sum of squares of those neighboring pixel values which overlap the filter placed over the pixel \( (x, y) \). The unnormalized square box filter can be useful in computing local image statistics such as the the local variance and standard deviation around the neighborhood of a pixel.Parameters:src - input imagedst - output image of the same size and type as \_srcddepth - the output image depth (-1 to use src.depth())ksize - kernel size center. SEE: boxFilter

#### sqrBoxFilter public static void sqrBoxFilter([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor) Calculates the normalized sum of squares of the pixel values overlapping the filter. For every pixel \( (x, y) \) in the source image, the function calculates the sum of squares of those neighboring pixel values which overlap the filter placed over the pixel \( (x, y) \). The unnormalized square box filter can be useful in computing local image statistics such as the the local variance and standard deviation around the neighborhood of a pixel.Parameters:src - input imagedst - output image of the same size and type as \_srcddepth - the output image depth (-1 to use src.depth())ksize - kernel sizeanchor - kernel anchor point. The default value of Point(-1, -1) denotes that the anchor is at the kernel center. SEE: boxFilter

#### sqrBoxFilter public static void sqrBoxFilter([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, boolean normalize) Calculates the normalized sum of squares of the pixel values overlapping the filter. For every pixel \( (x, y) \) in the source image, the function calculates the sum of squares of those neighboring pixel values which overlap the filter placed over the pixel \( (x, y) \). The unnormalized square box filter can be useful in computing local image statistics such as the the local variance and standard deviation around the neighborhood of a pixel.Parameters:src - input imagedst - output image of the same size and type as \_srcddepth - the output image depth (-1 to use src.depth())ksize - kernel sizeanchor - kernel anchor point. The default value of Point(-1, -1) denotes that the anchor is at the kernel center.normalize - flag, specifying whether the kernel is to be normalized by it's area or not. SEE: boxFilter

#### sqrBoxFilter public static void sqrBoxFilter([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int ddepth, [Size](http://docs.google.com/org/opencv/core/Size.html) ksize, [Point](http://docs.google.com/org/opencv/core/Point.html) anchor, boolean normalize, int borderType) Calculates the normalized sum of squares of the pixel values overlapping the filter. For every pixel \( (x, y) \) in the source image, the function calculates the sum of squares of those neighboring pixel values which overlap the filter placed over the pixel \( (x, y) \). The unnormalized square box filter can be useful in computing local image statistics such as the the local variance and standard deviation around the neighborhood of a pixel.Parameters:src - input imagedst - output image of the same size and type as \_srcddepth - the output image depth (-1 to use src.depth())ksize - kernel sizeanchor - kernel anchor point. The default value of Point(-1, -1) denotes that the anchor is at the kernel center.normalize - flag, specifying whether the kernel is to be normalized by it's area or not.borderType - border mode used to extrapolate pixels outside of the image, see #BorderTypes. #BORDER\_WRAP is not supported. SEE: boxFilter

#### threshold public static double threshold([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, double thresh, double maxval, int type) Applies a fixed-level threshold to each array element. The function applies fixed-level thresholding to a multiple-channel array. The function is typically used to get a bi-level (binary) image out of a grayscale image ( #compare could be also used for this purpose) or for removing a noise, that is, filtering out pixels with too small or too large values. There are several types of thresholding supported by the function. They are determined by type parameter. Also, the special values #THRESH\_OTSU or #THRESH\_TRIANGLE may be combined with one of the above values. In these cases, the function determines the optimal threshold value using the Otsu's or Triangle algorithm and uses it instead of the specified thresh. **Note:** Currently, the Otsu's and Triangle methods are implemented only for 8-bit single-channel images.Parameters:src - input array (multiple-channel, 8-bit or 32-bit floating point).dst - output array of the same size and type and the same number of channels as src.thresh - threshold value.maxval - maximum value to use with the #THRESH\_BINARY and #THRESH\_BINARY\_INV thresholding types.type - thresholding type (see #ThresholdTypes). Returns:the computed threshold value if Otsu's or Triangle methods used. SEE: adaptiveThreshold, findContours, compare, min, max

#### undistort public static void undistort([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs) Transforms an image to compensate for lens distortion. The function transforms an image to compensate radial and tangential lens distortion. The function is simply a combination of #initUndistortRectifyMap (with unity R ) and #remap (with bilinear interpolation). See the former function for details of the transformation being performed. Those pixels in the destination image, for which there is no correspondent pixels in the source image, are filled with zeros (black color). A particular subset of the source image that will be visible in the corrected image can be regulated by newCameraMatrix. You can use #getOptimalNewCameraMatrix to compute the appropriate newCameraMatrix depending on your requirements. The camera matrix and the distortion parameters can be determined using #calibrateCamera. If the resolution of images is different from the resolution used at the calibration stage, \(f\_x, f\_y, c\_x\) and \(c\_y\) need to be scaled accordingly, while the distortion coefficients remain the same.Parameters:src - Input (distorted) image.dst - Output (corrected) image that has the same size and type as src .cameraMatrix - Input camera matrix \(A = \vecthreethree{f\_x}{0}{c\_x}{0}{f\_y}{c\_y}{0}{0}{1}\) .distCoeffs - Input vector of distortion coefficients \((k\_1, k\_2, p\_1, p\_2[, k\_3[, k\_4, k\_5, k\_6[, s\_1, s\_2, s\_3, s\_4[, \tau\_x, \tau\_y]]]])\) of 4, 5, 8, 12 or 14 elements. If the vector is NULL/empty, the zero distortion coefficients are assumed. cameraMatrix but you may additionally scale and shift the result by using a different matrix.

#### undistort public static void undistort([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) newCameraMatrix) Transforms an image to compensate for lens distortion. The function transforms an image to compensate radial and tangential lens distortion. The function is simply a combination of #initUndistortRectifyMap (with unity R ) and #remap (with bilinear interpolation). See the former function for details of the transformation being performed. Those pixels in the destination image, for which there is no correspondent pixels in the source image, are filled with zeros (black color). A particular subset of the source image that will be visible in the corrected image can be regulated by newCameraMatrix. You can use #getOptimalNewCameraMatrix to compute the appropriate newCameraMatrix depending on your requirements. The camera matrix and the distortion parameters can be determined using #calibrateCamera. If the resolution of images is different from the resolution used at the calibration stage, \(f\_x, f\_y, c\_x\) and \(c\_y\) need to be scaled accordingly, while the distortion coefficients remain the same.Parameters:src - Input (distorted) image.dst - Output (corrected) image that has the same size and type as src .cameraMatrix - Input camera matrix \(A = \vecthreethree{f\_x}{0}{c\_x}{0}{f\_y}{c\_y}{0}{0}{1}\) .distCoeffs - Input vector of distortion coefficients \((k\_1, k\_2, p\_1, p\_2[, k\_3[, k\_4, k\_5, k\_6[, s\_1, s\_2, s\_3, s\_4[, \tau\_x, \tau\_y]]]])\) of 4, 5, 8, 12 or 14 elements. If the vector is NULL/empty, the zero distortion coefficients are assumed.newCameraMatrix - Camera matrix of the distorted image. By default, it is the same as cameraMatrix but you may additionally scale and shift the result by using a different matrix.

#### undistortPoints public static void undistortPoints([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs) Computes the ideal point coordinates from the observed point coordinates. The function is similar to #undistort and #initUndistortRectifyMap but it operates on a sparse set of points instead of a raster image. Also the function performs a reverse transformation to projectPoints. In case of a 3D object, it does not reconstruct its 3D coordinates, but for a planar object, it does, up to a translation vector, if the proper R is specified. For each observed point coordinate \((u, v)\) the function computes: \( \begin{array}{l} x^{"} \leftarrow (u - c\_x)/f\_x \\ y^{"} \leftarrow (v - c\_y)/f\_y \\ (x',y') = undistort(x^{"},y^{"}, \texttt{distCoeffs}) \\ {[X\,Y\,W]} ^T \leftarrow R\*[x' \, y' \, 1]^T \\ x \leftarrow X/W \\ y \leftarrow Y/W \\ \text{only performed if P is specified:} \\ u' \leftarrow x {f'}\_x + {c'}\_x \\ v' \leftarrow y {f'}\_y + {c'}\_y \end{array} \) where \*undistort\* is an approximate iterative algorithm that estimates the normalized original point coordinates out of the normalized distorted point coordinates ("normalized" means that the coordinates do not depend on the camera matrix). The function can be used for both a stereo camera head or a monocular camera (when R is empty).Parameters:src - Observed point coordinates, 2xN/Nx2 1-channel or 1xN/Nx1 2-channel (CV\_32FC2 or CV\_64FC2) (or vector<Point2f> ).dst - Output ideal point coordinates (1xN/Nx1 2-channel or vector<Point2f> ) after undistortion and reverse perspective transformation. If matrix P is identity or omitted, dst will contain normalized point coordinates.cameraMatrix - Camera matrix \(\vecthreethree{f\_x}{0}{c\_x}{0}{f\_y}{c\_y}{0}{0}{1}\) .distCoeffs - Input vector of distortion coefficients \((k\_1, k\_2, p\_1, p\_2[, k\_3[, k\_4, k\_5, k\_6[, s\_1, s\_2, s\_3, s\_4[, \tau\_x, \tau\_y]]]])\) of 4, 5, 8, 12 or 14 elements. If the vector is NULL/empty, the zero distortion coefficients are assumed. #stereoRectify can be passed here. If the matrix is empty, the identity transformation is used. #stereoRectify can be passed here. If the matrix is empty, the identity new camera matrix is used.

#### undistortPoints public static void undistortPoints([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) R) Computes the ideal point coordinates from the observed point coordinates. The function is similar to #undistort and #initUndistortRectifyMap but it operates on a sparse set of points instead of a raster image. Also the function performs a reverse transformation to projectPoints. In case of a 3D object, it does not reconstruct its 3D coordinates, but for a planar object, it does, up to a translation vector, if the proper R is specified. For each observed point coordinate \((u, v)\) the function computes: \( \begin{array}{l} x^{"} \leftarrow (u - c\_x)/f\_x \\ y^{"} \leftarrow (v - c\_y)/f\_y \\ (x',y') = undistort(x^{"},y^{"}, \texttt{distCoeffs}) \\ {[X\,Y\,W]} ^T \leftarrow R\*[x' \, y' \, 1]^T \\ x \leftarrow X/W \\ y \leftarrow Y/W \\ \text{only performed if P is specified:} \\ u' \leftarrow x {f'}\_x + {c'}\_x \\ v' \leftarrow y {f'}\_y + {c'}\_y \end{array} \) where \*undistort\* is an approximate iterative algorithm that estimates the normalized original point coordinates out of the normalized distorted point coordinates ("normalized" means that the coordinates do not depend on the camera matrix). The function can be used for both a stereo camera head or a monocular camera (when R is empty).Parameters:src - Observed point coordinates, 2xN/Nx2 1-channel or 1xN/Nx1 2-channel (CV\_32FC2 or CV\_64FC2) (or vector<Point2f> ).dst - Output ideal point coordinates (1xN/Nx1 2-channel or vector<Point2f> ) after undistortion and reverse perspective transformation. If matrix P is identity or omitted, dst will contain normalized point coordinates.cameraMatrix - Camera matrix \(\vecthreethree{f\_x}{0}{c\_x}{0}{f\_y}{c\_y}{0}{0}{1}\) .distCoeffs - Input vector of distortion coefficients \((k\_1, k\_2, p\_1, p\_2[, k\_3[, k\_4, k\_5, k\_6[, s\_1, s\_2, s\_3, s\_4[, \tau\_x, \tau\_y]]]])\) of 4, 5, 8, 12 or 14 elements. If the vector is NULL/empty, the zero distortion coefficients are assumed.R - Rectification transformation in the object space (3x3 matrix). R1 or R2 computed by #stereoRectify can be passed here. If the matrix is empty, the identity transformation is used. #stereoRectify can be passed here. If the matrix is empty, the identity new camera matrix is used.

#### undistortPoints public static void undistortPoints([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) R, [Mat](http://docs.google.com/org/opencv/core/Mat.html) P) Computes the ideal point coordinates from the observed point coordinates. The function is similar to #undistort and #initUndistortRectifyMap but it operates on a sparse set of points instead of a raster image. Also the function performs a reverse transformation to projectPoints. In case of a 3D object, it does not reconstruct its 3D coordinates, but for a planar object, it does, up to a translation vector, if the proper R is specified. For each observed point coordinate \((u, v)\) the function computes: \( \begin{array}{l} x^{"} \leftarrow (u - c\_x)/f\_x \\ y^{"} \leftarrow (v - c\_y)/f\_y \\ (x',y') = undistort(x^{"},y^{"}, \texttt{distCoeffs}) \\ {[X\,Y\,W]} ^T \leftarrow R\*[x' \, y' \, 1]^T \\ x \leftarrow X/W \\ y \leftarrow Y/W \\ \text{only performed if P is specified:} \\ u' \leftarrow x {f'}\_x + {c'}\_x \\ v' \leftarrow y {f'}\_y + {c'}\_y \end{array} \) where \*undistort\* is an approximate iterative algorithm that estimates the normalized original point coordinates out of the normalized distorted point coordinates ("normalized" means that the coordinates do not depend on the camera matrix). The function can be used for both a stereo camera head or a monocular camera (when R is empty).Parameters:src - Observed point coordinates, 2xN/Nx2 1-channel or 1xN/Nx1 2-channel (CV\_32FC2 or CV\_64FC2) (or vector<Point2f> ).dst - Output ideal point coordinates (1xN/Nx1 2-channel or vector<Point2f> ) after undistortion and reverse perspective transformation. If matrix P is identity or omitted, dst will contain normalized point coordinates.cameraMatrix - Camera matrix \(\vecthreethree{f\_x}{0}{c\_x}{0}{f\_y}{c\_y}{0}{0}{1}\) .distCoeffs - Input vector of distortion coefficients \((k\_1, k\_2, p\_1, p\_2[, k\_3[, k\_4, k\_5, k\_6[, s\_1, s\_2, s\_3, s\_4[, \tau\_x, \tau\_y]]]])\) of 4, 5, 8, 12 or 14 elements. If the vector is NULL/empty, the zero distortion coefficients are assumed.R - Rectification transformation in the object space (3x3 matrix). R1 or R2 computed by #stereoRectify can be passed here. If the matrix is empty, the identity transformation is used.P - New camera matrix (3x3) or new projection matrix (3x4) \(\begin{bmatrix} {f'}\_x & 0 & {c'}\_x & t\_x \\ 0 & {f'}\_y & {c'}\_y & t\_y \\ 0 & 0 & 1 & t\_z \end{bmatrix}\). P1 or P2 computed by #stereoRectify can be passed here. If the matrix is empty, the identity new camera matrix is used.

#### undistortPointsIter public static void undistortPointsIter([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) cameraMatrix, [Mat](http://docs.google.com/org/opencv/core/Mat.html) distCoeffs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) R, [Mat](http://docs.google.com/org/opencv/core/Mat.html) P, [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) criteria) **Note:** Default version of #undistortPoints does 5 iterations to compute undistorted points.Parameters:src - automatically generateddst - automatically generatedcameraMatrix - automatically generateddistCoeffs - automatically generatedR - automatically generatedP - automatically generatedcriteria - automatically generated

#### warpAffine public static void warpAffine([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) M, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize) Applies an affine transformation to an image. The function warpAffine transforms the source image using the specified matrix: \(\texttt{dst} (x,y) = \texttt{src} ( \texttt{M} \_{11} x + \texttt{M} \_{12} y + \texttt{M} \_{13}, \texttt{M} \_{21} x + \texttt{M} \_{22} y + \texttt{M} \_{23})\) when the flag #WARP\_INVERSE\_MAP is set. Otherwise, the transformation is first inverted with #invertAffineTransform and then put in the formula above instead of M. The function cannot operate in-place.Parameters:src - input image.dst - output image that has the size dsize and the same type as src .M - \(2\times 3\) transformation matrix.dsize - size of the output image. flag #WARP\_INVERSE\_MAP that means that M is the inverse transformation ( \(\texttt{dst}\rightarrow\texttt{src}\) ). borderMode=#BORDER\_TRANSPARENT, it means that the pixels in the destination image corresponding to the "outliers" in the source image are not modified by the function. SEE: warpPerspective, resize, remap, getRectSubPix, transform

#### warpAffine public static void warpAffine([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) M, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, int flags) Applies an affine transformation to an image. The function warpAffine transforms the source image using the specified matrix: \(\texttt{dst} (x,y) = \texttt{src} ( \texttt{M} \_{11} x + \texttt{M} \_{12} y + \texttt{M} \_{13}, \texttt{M} \_{21} x + \texttt{M} \_{22} y + \texttt{M} \_{23})\) when the flag #WARP\_INVERSE\_MAP is set. Otherwise, the transformation is first inverted with #invertAffineTransform and then put in the formula above instead of M. The function cannot operate in-place.Parameters:src - input image.dst - output image that has the size dsize and the same type as src .M - \(2\times 3\) transformation matrix.dsize - size of the output image.flags - combination of interpolation methods (see #InterpolationFlags) and the optional flag #WARP\_INVERSE\_MAP that means that M is the inverse transformation ( \(\texttt{dst}\rightarrow\texttt{src}\) ). borderMode=#BORDER\_TRANSPARENT, it means that the pixels in the destination image corresponding to the "outliers" in the source image are not modified by the function. SEE: warpPerspective, resize, remap, getRectSubPix, transform

#### warpAffine public static void warpAffine([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) M, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, int flags, int borderMode) Applies an affine transformation to an image. The function warpAffine transforms the source image using the specified matrix: \(\texttt{dst} (x,y) = \texttt{src} ( \texttt{M} \_{11} x + \texttt{M} \_{12} y + \texttt{M} \_{13}, \texttt{M} \_{21} x + \texttt{M} \_{22} y + \texttt{M} \_{23})\) when the flag #WARP\_INVERSE\_MAP is set. Otherwise, the transformation is first inverted with #invertAffineTransform and then put in the formula above instead of M. The function cannot operate in-place.Parameters:src - input image.dst - output image that has the size dsize and the same type as src .M - \(2\times 3\) transformation matrix.dsize - size of the output image.flags - combination of interpolation methods (see #InterpolationFlags) and the optional flag #WARP\_INVERSE\_MAP that means that M is the inverse transformation ( \(\texttt{dst}\rightarrow\texttt{src}\) ).borderMode - pixel extrapolation method (see #BorderTypes); when borderMode=#BORDER\_TRANSPARENT, it means that the pixels in the destination image corresponding to the "outliers" in the source image are not modified by the function. SEE: warpPerspective, resize, remap, getRectSubPix, transform

#### warpAffine public static void warpAffine([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) M, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, int flags, int borderMode, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) borderValue) Applies an affine transformation to an image. The function warpAffine transforms the source image using the specified matrix: \(\texttt{dst} (x,y) = \texttt{src} ( \texttt{M} \_{11} x + \texttt{M} \_{12} y + \texttt{M} \_{13}, \texttt{M} \_{21} x + \texttt{M} \_{22} y + \texttt{M} \_{23})\) when the flag #WARP\_INVERSE\_MAP is set. Otherwise, the transformation is first inverted with #invertAffineTransform and then put in the formula above instead of M. The function cannot operate in-place.Parameters:src - input image.dst - output image that has the size dsize and the same type as src .M - \(2\times 3\) transformation matrix.dsize - size of the output image.flags - combination of interpolation methods (see #InterpolationFlags) and the optional flag #WARP\_INVERSE\_MAP that means that M is the inverse transformation ( \(\texttt{dst}\rightarrow\texttt{src}\) ).borderMode - pixel extrapolation method (see #BorderTypes); when borderMode=#BORDER\_TRANSPARENT, it means that the pixels in the destination image corresponding to the "outliers" in the source image are not modified by the function.borderValue - value used in case of a constant border; by default, it is 0. SEE: warpPerspective, resize, remap, getRectSubPix, transform

#### warpPerspective public static void warpPerspective([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) M, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize) Applies a perspective transformation to an image. The function warpPerspective transforms the source image using the specified matrix: \(\texttt{dst} (x,y) = \texttt{src} \left ( \frac{M\_{11} x + M\_{12} y + M\_{13}}{M\_{31} x + M\_{32} y + M\_{33}} , \frac{M\_{21} x + M\_{22} y + M\_{23}}{M\_{31} x + M\_{32} y + M\_{33}} \right )\) when the flag #WARP\_INVERSE\_MAP is set. Otherwise, the transformation is first inverted with invert and then put in the formula above instead of M. The function cannot operate in-place.Parameters:src - input image.dst - output image that has the size dsize and the same type as src .M - \(3\times 3\) transformation matrix.dsize - size of the output image. optional flag #WARP\_INVERSE\_MAP, that sets M as the inverse transformation ( \(\texttt{dst}\rightarrow\texttt{src}\) ). SEE: warpAffine, resize, remap, getRectSubPix, perspectiveTransform

#### warpPerspective public static void warpPerspective([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) M, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, int flags) Applies a perspective transformation to an image. The function warpPerspective transforms the source image using the specified matrix: \(\texttt{dst} (x,y) = \texttt{src} \left ( \frac{M\_{11} x + M\_{12} y + M\_{13}}{M\_{31} x + M\_{32} y + M\_{33}} , \frac{M\_{21} x + M\_{22} y + M\_{23}}{M\_{31} x + M\_{32} y + M\_{33}} \right )\) when the flag #WARP\_INVERSE\_MAP is set. Otherwise, the transformation is first inverted with invert and then put in the formula above instead of M. The function cannot operate in-place.Parameters:src - input image.dst - output image that has the size dsize and the same type as src .M - \(3\times 3\) transformation matrix.dsize - size of the output image.flags - combination of interpolation methods (#INTER\_LINEAR or #INTER\_NEAREST) and the optional flag #WARP\_INVERSE\_MAP, that sets M as the inverse transformation ( \(\texttt{dst}\rightarrow\texttt{src}\) ). SEE: warpAffine, resize, remap, getRectSubPix, perspectiveTransform

#### warpPerspective public static void warpPerspective([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) M, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, int flags, int borderMode) Applies a perspective transformation to an image. The function warpPerspective transforms the source image using the specified matrix: \(\texttt{dst} (x,y) = \texttt{src} \left ( \frac{M\_{11} x + M\_{12} y + M\_{13}}{M\_{31} x + M\_{32} y + M\_{33}} , \frac{M\_{21} x + M\_{22} y + M\_{23}}{M\_{31} x + M\_{32} y + M\_{33}} \right )\) when the flag #WARP\_INVERSE\_MAP is set. Otherwise, the transformation is first inverted with invert and then put in the formula above instead of M. The function cannot operate in-place.Parameters:src - input image.dst - output image that has the size dsize and the same type as src .M - \(3\times 3\) transformation matrix.dsize - size of the output image.flags - combination of interpolation methods (#INTER\_LINEAR or #INTER\_NEAREST) and the optional flag #WARP\_INVERSE\_MAP, that sets M as the inverse transformation ( \(\texttt{dst}\rightarrow\texttt{src}\) ).borderMode - pixel extrapolation method (#BORDER\_CONSTANT or #BORDER\_REPLICATE). SEE: warpAffine, resize, remap, getRectSubPix, perspectiveTransform

#### warpPerspective public static void warpPerspective([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Mat](http://docs.google.com/org/opencv/core/Mat.html) M, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, int flags, int borderMode, [Scalar](http://docs.google.com/org/opencv/core/Scalar.html) borderValue) Applies a perspective transformation to an image. The function warpPerspective transforms the source image using the specified matrix: \(\texttt{dst} (x,y) = \texttt{src} \left ( \frac{M\_{11} x + M\_{12} y + M\_{13}}{M\_{31} x + M\_{32} y + M\_{33}} , \frac{M\_{21} x + M\_{22} y + M\_{23}}{M\_{31} x + M\_{32} y + M\_{33}} \right )\) when the flag #WARP\_INVERSE\_MAP is set. Otherwise, the transformation is first inverted with invert and then put in the formula above instead of M. The function cannot operate in-place.Parameters:src - input image.dst - output image that has the size dsize and the same type as src .M - \(3\times 3\) transformation matrix.dsize - size of the output image.flags - combination of interpolation methods (#INTER\_LINEAR or #INTER\_NEAREST) and the optional flag #WARP\_INVERSE\_MAP, that sets M as the inverse transformation ( \(\texttt{dst}\rightarrow\texttt{src}\) ).borderMode - pixel extrapolation method (#BORDER\_CONSTANT or #BORDER\_REPLICATE).borderValue - value used in case of a constant border; by default, it equals 0. SEE: warpAffine, resize, remap, getRectSubPix, perspectiveTransform

#### warpPolar public static void warpPolar([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [Size](http://docs.google.com/org/opencv/core/Size.html) dsize, [Point](http://docs.google.com/org/opencv/core/Point.html) center, double maxRadius, int flags)

Remaps an image to polar or semilog-polar coordinates space polar\_remaps\_reference\_image ![Polar remaps reference](pics/polar\_remap\_doc.png) Transform the source image using the following transformation: \( dst(\rho , \phi ) = src(x,y) \) where \( \begin{array}{l} \vec{I} = (x - center.x, \;y - center.y) \\ \phi = Kangle \cdot \texttt{angle} (\vec{I}) \\ \rho = \left\{\begin{matrix} Klin \cdot \texttt{magnitude} (\vec{I}) & default \\ Klog \cdot log\_e(\texttt{magnitude} (\vec{I})) & if \; semilog \\ \end{matrix}\right. \end{array} \) and \( \begin{array}{l} Kangle = dsize.height / 2\Pi \\ Klin = dsize.width / maxRadius \\ Klog = dsize.width / log\_e(maxRadius) \\ \end{array} \) \par Linear vs semilog mapping Polar mapping can be linear or semi-log. Add one of #WarpPolarMode to flags to specify the polar mapping mode. Linear is the default mode. The semilog mapping emulates the human "foveal" vision that permit very high acuity on the line of sight (central vision) in contrast to peripheral vision where acuity is minor. \par Option on dsize:

* + - * if both values in dsize <=0 (default), the destination image will have (almost) same area of source bounding circle: \(\begin{array}{l} dsize.area \leftarrow (maxRadius^2 \cdot \Pi) \\ dsize.width = \texttt{cvRound}(maxRadius) \\ dsize.height = \texttt{cvRound}(maxRadius \cdot \Pi) \\ \end{array}\)
      * if only dsize.height <= 0, the destination image area will be proportional to the bounding circle area but scaled by Kx \* Kx: \(\begin{array}{l} dsize.height = \texttt{cvRound}(dsize.width \cdot \Pi) \\ \end{array} \)
      * if both values in dsize > 0 , the destination image will have the given size therefore the area of the bounding circle will be scaled to dsize.

\par Reverse mapping You can get reverse mapping adding #WARP\_INVERSE\_MAP to flags \snippet polar\_transforms.cpp InverseMap In addiction, to calculate the original coordinate from a polar mapped coordinate \((rho, phi)->(x, y)\): \snippet polar\_transforms.cpp InverseCoordinateParameters:src - Source image.dst - Destination image. It will have same type as src.dsize - The destination image size (see description for valid options).center - The transformation center.maxRadius - The radius of the bounding circle to transform. It determines the inverse magnitude scale parameter too.flags - A combination of interpolation methods, #InterpolationFlags + #WarpPolarMode.

* + - * Add #WARP\_POLAR\_LINEAR to select linear polar mapping (default)
      * Add #WARP\_POLAR\_LOG to select semilog polar mapping
      * Add #WARP\_INVERSE\_MAP for reverse mapping.

**Note:**

* + - * The function can not operate in-place.
      * To calculate magnitude and angle in degrees #cartToPolar is used internally thus angles are measured from 0 to 360 with accuracy about 0.3 degrees.
      * This function uses #remap. Due to current implementation limitations the size of an input and output images should be less than 32767x32767.

SEE: cv::remap

#### watershed public static void watershed([Mat](http://docs.google.com/org/opencv/core/Mat.html) image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) markers) Performs a marker-based image segmentation using the watershed algorithm. The function implements one of the variants of watershed, non-parametric marker-based segmentation algorithm, described in CITE: Meyer92 . Before passing the image to the function, you have to roughly outline the desired regions in the image markers with positive (>0) indices. So, every region is represented as one or more connected components with the pixel values 1, 2, 3, and so on. Such markers can be retrieved from a binary mask using #findContours and #drawContours (see the watershed.cpp demo). The markers are "seeds" of the future image regions. All the other pixels in markers , whose relation to the outlined regions is not known and should be defined by the algorithm, should be set to 0's. In the function output, each pixel in markers is set to a value of the "seed" components or to -1 at boundaries between the regions. **Note:** Any two neighbor connected components are not necessarily separated by a watershed boundary (-1's pixels); for example, they can touch each other in the initial marker image passed to the function.Parameters:image - Input 8-bit 3-channel image.markers - Input/output 32-bit single-channel image (map) of markers. It should have the same size as image . SEE: findContours imgproc\_misc

* [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/imgproc/GeneralizedHoughGuil.html)
* [Next Class](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html)
* [Frames](http://docs.google.com/index.html?org/opencv/imgproc/Imgproc.html)
* [No Frames](http://docs.google.com/Imgproc.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* [Field](#3znysh7) |
* [Constr](#2et92p0) |
* [Method](#tyjcwt)
* Detail:
* [Field](#1t3h5sf) |
* [Constr](#1pr1rn1) |
* [Method](#2ovzkin)

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