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/photo/MergeRobertson.html)
* [Next Class](http://docs.google.com/org/opencv/photo/Tonemap.html)
* [Frames](http://docs.google.com/index.html?org/opencv/photo/Photo.html)
* [No Frames](http://docs.google.com/Photo.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* [Field](#3znysh7) |
* [Constr](#2et92p0) |
* [Method](#tyjcwt)
* Detail:
* [Field](#1t3h5sf) |
* [Constr](#44sinio) |
* [Method](#z337ya)

org.opencv.photo

## Class Photo

* java.lang.Object
  + org.opencv.photo.Photo
* public class Photo  
  extends java.lang.Object

### Field SummaryFields

| Modifier and Type | Field and Description |
| --- | --- |
| static int | [**INPAINT\_NS**](http://docs.google.com/org/opencv/photo/Photo.html#INPAINT_NS) |
| static int | [**INPAINT\_TELEA**](http://docs.google.com/org/opencv/photo/Photo.html#INPAINT_TELEA) |
| static int | [**LDR\_SIZE**](http://docs.google.com/org/opencv/photo/Photo.html#LDR_SIZE) |
| static int | [**MIXED\_CLONE**](http://docs.google.com/org/opencv/photo/Photo.html#MIXED_CLONE) |
| static int | [**MONOCHROME\_TRANSFER**](http://docs.google.com/org/opencv/photo/Photo.html#MONOCHROME_TRANSFER) |
| static int | [**NORMAL\_CLONE**](http://docs.google.com/org/opencv/photo/Photo.html#NORMAL_CLONE) |
| static int | [**NORMCONV\_FILTER**](http://docs.google.com/org/opencv/photo/Photo.html#NORMCONV_FILTER) |
| static int | [**RECURS\_FILTER**](http://docs.google.com/org/opencv/photo/Photo.html#RECURS_FILTER) |

### Constructor SummaryConstructors

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

### Method SummaryMethods

| Modifier and Type | Method and Description |
| --- | --- |
| static void | [**colorChange**](http://docs.google.com/org/opencv/photo/Photo.html#colorChange(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) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Given an original color image, two differently colored versions of this image can be mixed seamlessly. |
| static void | [**colorChange**](http://docs.google.com/org/opencv/photo/Photo.html#colorChange(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20float))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float red\_mul) Given an original color image, two differently colored versions of this image can be mixed seamlessly. |
| static void | [**colorChange**](http://docs.google.com/org/opencv/photo/Photo.html#colorChange(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20float,%20float))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float red\_mul, float green\_mul) Given an original color image, two differently colored versions of this image can be mixed seamlessly. |
| static void | [**colorChange**](http://docs.google.com/org/opencv/photo/Photo.html#colorChange(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20float,%20float,%20float))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float red\_mul, float green\_mul, float blue\_mul) Given an original color image, two differently colored versions of this image can be mixed seamlessly. |
| static [AlignMTB](http://docs.google.com/org/opencv/photo/AlignMTB.html) | [**createAlignMTB**](http://docs.google.com/org/opencv/photo/Photo.html#createAlignMTB())() Creates AlignMTB object usually good enough (31 and 63 pixels shift respectively). |
| static [AlignMTB](http://docs.google.com/org/opencv/photo/AlignMTB.html) | [**createAlignMTB**](http://docs.google.com/org/opencv/photo/Photo.html#createAlignMTB(int))(int max\_bits) Creates AlignMTB object |
| static [AlignMTB](http://docs.google.com/org/opencv/photo/AlignMTB.html) | [**createAlignMTB**](http://docs.google.com/org/opencv/photo/Photo.html#createAlignMTB(int,%20int))(int max\_bits, int exclude\_range) Creates AlignMTB object |
| static [AlignMTB](http://docs.google.com/org/opencv/photo/AlignMTB.html) | [**createAlignMTB**](http://docs.google.com/org/opencv/photo/Photo.html#createAlignMTB(int,%20int,%20boolean))(int max\_bits, int exclude\_range, boolean cut) Creates AlignMTB object |
| static [CalibrateDebevec](http://docs.google.com/org/opencv/photo/CalibrateDebevec.html) | [**createCalibrateDebevec**](http://docs.google.com/org/opencv/photo/Photo.html#createCalibrateDebevec())() Creates CalibrateDebevec object response. |
| static [CalibrateDebevec](http://docs.google.com/org/opencv/photo/CalibrateDebevec.html) | [**createCalibrateDebevec**](http://docs.google.com/org/opencv/photo/Photo.html#createCalibrateDebevec(int))(int samples) Creates CalibrateDebevec object |
| static [CalibrateDebevec](http://docs.google.com/org/opencv/photo/CalibrateDebevec.html) | [**createCalibrateDebevec**](http://docs.google.com/org/opencv/photo/Photo.html#createCalibrateDebevec(int,%20float))(int samples, float lambda) Creates CalibrateDebevec object |
| static [CalibrateDebevec](http://docs.google.com/org/opencv/photo/CalibrateDebevec.html) | [**createCalibrateDebevec**](http://docs.google.com/org/opencv/photo/Photo.html#createCalibrateDebevec(int,%20float,%20boolean))(int samples, float lambda, boolean random) Creates CalibrateDebevec object |
| static [CalibrateRobertson](http://docs.google.com/org/opencv/photo/CalibrateRobertson.html) | [**createCalibrateRobertson**](http://docs.google.com/org/opencv/photo/Photo.html#createCalibrateRobertson())() Creates CalibrateRobertson object |
| static [CalibrateRobertson](http://docs.google.com/org/opencv/photo/CalibrateRobertson.html) | [**createCalibrateRobertson**](http://docs.google.com/org/opencv/photo/Photo.html#createCalibrateRobertson(int))(int max\_iter) Creates CalibrateRobertson object |
| static [CalibrateRobertson](http://docs.google.com/org/opencv/photo/CalibrateRobertson.html) | [**createCalibrateRobertson**](http://docs.google.com/org/opencv/photo/Photo.html#createCalibrateRobertson(int,%20float))(int max\_iter, float threshold) Creates CalibrateRobertson object |
| static [MergeDebevec](http://docs.google.com/org/opencv/photo/MergeDebevec.html) | [**createMergeDebevec**](http://docs.google.com/org/opencv/photo/Photo.html#createMergeDebevec())() Creates MergeDebevec object |
| static [MergeMertens](http://docs.google.com/org/opencv/photo/MergeMertens.html) | [**createMergeMertens**](http://docs.google.com/org/opencv/photo/Photo.html#createMergeMertens())() Creates MergeMertens object |
| static [MergeMertens](http://docs.google.com/org/opencv/photo/MergeMertens.html) | [**createMergeMertens**](http://docs.google.com/org/opencv/photo/Photo.html#createMergeMertens(float))(float contrast\_weight) Creates MergeMertens object |
| static [MergeMertens](http://docs.google.com/org/opencv/photo/MergeMertens.html) | [**createMergeMertens**](http://docs.google.com/org/opencv/photo/Photo.html#createMergeMertens(float,%20float))(float contrast\_weight, float saturation\_weight) Creates MergeMertens object |
| static [MergeMertens](http://docs.google.com/org/opencv/photo/MergeMertens.html) | [**createMergeMertens**](http://docs.google.com/org/opencv/photo/Photo.html#createMergeMertens(float,%20float,%20float))(float contrast\_weight, float saturation\_weight, float exposure\_weight) Creates MergeMertens object |
| static [MergeRobertson](http://docs.google.com/org/opencv/photo/MergeRobertson.html) | [**createMergeRobertson**](http://docs.google.com/org/opencv/photo/Photo.html#createMergeRobertson())() Creates MergeRobertson object |
| static [Tonemap](http://docs.google.com/org/opencv/photo/Tonemap.html) | [**createTonemap**](http://docs.google.com/org/opencv/photo/Photo.html#createTonemap())() Creates simple linear mapper with gamma correction equal to 2.2f is suitable for most displays. |
| static [Tonemap](http://docs.google.com/org/opencv/photo/Tonemap.html) | [**createTonemap**](http://docs.google.com/org/opencv/photo/Photo.html#createTonemap(float))(float gamma) Creates simple linear mapper with gamma correction |
| static [TonemapDrago](http://docs.google.com/org/opencv/photo/TonemapDrago.html) | [**createTonemapDrago**](http://docs.google.com/org/opencv/photo/Photo.html#createTonemapDrago())() Creates TonemapDrago object than 1 increase saturation and values less than 1 decrease it. |
| static [TonemapDrago](http://docs.google.com/org/opencv/photo/TonemapDrago.html) | [**createTonemapDrago**](http://docs.google.com/org/opencv/photo/Photo.html#createTonemapDrago(float))(float gamma) Creates TonemapDrago object |
| static [TonemapDrago](http://docs.google.com/org/opencv/photo/TonemapDrago.html) | [**createTonemapDrago**](http://docs.google.com/org/opencv/photo/Photo.html#createTonemapDrago(float,%20float))(float gamma, float saturation) Creates TonemapDrago object |
| static [TonemapDrago](http://docs.google.com/org/opencv/photo/TonemapDrago.html) | [**createTonemapDrago**](http://docs.google.com/org/opencv/photo/Photo.html#createTonemapDrago(float,%20float,%20float))(float gamma, float saturation, float bias) Creates TonemapDrago object |
| static [TonemapMantiuk](http://docs.google.com/org/opencv/photo/TonemapMantiuk.html) | [**createTonemapMantiuk**](http://docs.google.com/org/opencv/photo/Photo.html#createTonemapMantiuk())() Creates TonemapMantiuk object dynamic range. |
| static [TonemapMantiuk](http://docs.google.com/org/opencv/photo/TonemapMantiuk.html) | [**createTonemapMantiuk**](http://docs.google.com/org/opencv/photo/Photo.html#createTonemapMantiuk(float))(float gamma) Creates TonemapMantiuk object |
| static [TonemapMantiuk](http://docs.google.com/org/opencv/photo/TonemapMantiuk.html) | [**createTonemapMantiuk**](http://docs.google.com/org/opencv/photo/Photo.html#createTonemapMantiuk(float,%20float))(float gamma, float scale) Creates TonemapMantiuk object |
| static [TonemapMantiuk](http://docs.google.com/org/opencv/photo/TonemapMantiuk.html) | [**createTonemapMantiuk**](http://docs.google.com/org/opencv/photo/Photo.html#createTonemapMantiuk(float,%20float,%20float))(float gamma, float scale, float saturation) Creates TonemapMantiuk object |
| static [TonemapReinhard](http://docs.google.com/org/opencv/photo/TonemapReinhard.html) | [**createTonemapReinhard**](http://docs.google.com/org/opencv/photo/Photo.html#createTonemapReinhard())() Creates TonemapReinhard object value, if 0 it's global, otherwise it's a weighted mean of this two cases. |
| static [TonemapReinhard](http://docs.google.com/org/opencv/photo/TonemapReinhard.html) | [**createTonemapReinhard**](http://docs.google.com/org/opencv/photo/Photo.html#createTonemapReinhard(float))(float gamma) Creates TonemapReinhard object |
| static [TonemapReinhard](http://docs.google.com/org/opencv/photo/TonemapReinhard.html) | [**createTonemapReinhard**](http://docs.google.com/org/opencv/photo/Photo.html#createTonemapReinhard(float,%20float))(float gamma, float intensity) Creates TonemapReinhard object |
| static [TonemapReinhard](http://docs.google.com/org/opencv/photo/TonemapReinhard.html) | [**createTonemapReinhard**](http://docs.google.com/org/opencv/photo/Photo.html#createTonemapReinhard(float,%20float,%20float))(float gamma, float intensity, float light\_adapt) Creates TonemapReinhard object |
| static [TonemapReinhard](http://docs.google.com/org/opencv/photo/TonemapReinhard.html) | [**createTonemapReinhard**](http://docs.google.com/org/opencv/photo/Photo.html#createTonemapReinhard(float,%20float,%20float,%20float))(float gamma, float intensity, float light\_adapt, float color\_adapt) Creates TonemapReinhard object |
| static void | [**decolor**](http://docs.google.com/org/opencv/photo/Photo.html#decolor(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) grayscale, [Mat](http://docs.google.com/org/opencv/core/Mat.html) color\_boost) Transforms a color image to a grayscale image. |
| static void | [**denoise\_TVL1**](http://docs.google.com/org/opencv/photo/Photo.html#denoise_TVL1(java.util.List,%20org.opencv.core.Mat))(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> observations, [Mat](http://docs.google.com/org/opencv/core/Mat.html) result) Primal-dual algorithm is an algorithm for solving special types of variational problems (that is, finding a function to minimize some functional). |
| static void | [**denoise\_TVL1**](http://docs.google.com/org/opencv/photo/Photo.html#denoise_TVL1(java.util.List,%20org.opencv.core.Mat,%20double))(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> observations, [Mat](http://docs.google.com/org/opencv/core/Mat.html) result, double lambda) Primal-dual algorithm is an algorithm for solving special types of variational problems (that is, finding a function to minimize some functional). |
| static void | [**denoise\_TVL1**](http://docs.google.com/org/opencv/photo/Photo.html#denoise_TVL1(java.util.List,%20org.opencv.core.Mat,%20double,%20int))(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> observations, [Mat](http://docs.google.com/org/opencv/core/Mat.html) result, double lambda, int niters) Primal-dual algorithm is an algorithm for solving special types of variational problems (that is, finding a function to minimize some functional). |
| static void | [**detailEnhance**](http://docs.google.com/org/opencv/photo/Photo.html#detailEnhance(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) This filter enhances the details of a particular image. |
| static void | [**detailEnhance**](http://docs.google.com/org/opencv/photo/Photo.html#detailEnhance(org.opencv.core.Mat,%20org.opencv.core.Mat,%20float))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float sigma\_s) This filter enhances the details of a particular image. |
| static void | [**detailEnhance**](http://docs.google.com/org/opencv/photo/Photo.html#detailEnhance(org.opencv.core.Mat,%20org.opencv.core.Mat,%20float,%20float))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float sigma\_s, float sigma\_r) This filter enhances the details of a particular image. |
| static void | [**edgePreservingFilter**](http://docs.google.com/org/opencv/photo/Photo.html#edgePreservingFilter(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) Filtering is the fundamental operation in image and video processing. |
| static void | [**edgePreservingFilter**](http://docs.google.com/org/opencv/photo/Photo.html#edgePreservingFilter(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 flags) Filtering is the fundamental operation in image and video processing. |
| static void | [**edgePreservingFilter**](http://docs.google.com/org/opencv/photo/Photo.html#edgePreservingFilter(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20float))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int flags, float sigma\_s) Filtering is the fundamental operation in image and video processing. |
| static void | [**edgePreservingFilter**](http://docs.google.com/org/opencv/photo/Photo.html#edgePreservingFilter(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int,%20float,%20float))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int flags, float sigma\_s, float sigma\_r) Filtering is the fundamental operation in image and video processing. |
| static void | [**fastNlMeansDenoising**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoising(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) Perform image denoising using Non-local Means Denoising algorithm <http://www.ipol.im/pub/algo/bcm\_non\_local\_means\_denoising/> with several computational optimizations. |
| static void | [**fastNlMeansDenoising**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoising(org.opencv.core.Mat,%20org.opencv.core.Mat,%20float))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float h) Perform image denoising using Non-local Means Denoising algorithm <http://www.ipol.im/pub/algo/bcm\_non\_local\_means\_denoising/> with several computational optimizations. |
| static void | [**fastNlMeansDenoising**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoising(org.opencv.core.Mat,%20org.opencv.core.Mat,%20float,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float h, int templateWindowSize) Perform image denoising using Non-local Means Denoising algorithm <http://www.ipol.im/pub/algo/bcm\_non\_local\_means\_denoising/> with several computational optimizations. |
| static void | [**fastNlMeansDenoising**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoising(org.opencv.core.Mat,%20org.opencv.core.Mat,%20float,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float h, int templateWindowSize, int searchWindowSize) Perform image denoising using Non-local Means Denoising algorithm <http://www.ipol.im/pub/algo/bcm\_non\_local\_means\_denoising/> with several computational optimizations. |
| static void | [**fastNlMeansDenoising**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoising(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.MatOfFloat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) h) Perform image denoising using Non-local Means Denoising algorithm <http://www.ipol.im/pub/algo/bcm\_non\_local\_means\_denoising/> with several computational optimizations. |
| static void | [**fastNlMeansDenoising**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoising(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.MatOfFloat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) h, int templateWindowSize) Perform image denoising using Non-local Means Denoising algorithm <http://www.ipol.im/pub/algo/bcm\_non\_local\_means\_denoising/> with several computational optimizations. |
| static void | [**fastNlMeansDenoising**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoising(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.MatOfFloat,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) h, int templateWindowSize, int searchWindowSize) Perform image denoising using Non-local Means Denoising algorithm <http://www.ipol.im/pub/algo/bcm\_non\_local\_means\_denoising/> with several computational optimizations. |
| static void | [**fastNlMeansDenoising**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoising(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.MatOfFloat,%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, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) h, int templateWindowSize, int searchWindowSize, int normType) Perform image denoising using Non-local Means Denoising algorithm <http://www.ipol.im/pub/algo/bcm\_non\_local\_means\_denoising/> with several computational optimizations. |
| static void | [**fastNlMeansDenoisingColored**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoisingColored(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) Modification of fastNlMeansDenoising function for colored images |
| static void | [**fastNlMeansDenoisingColored**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoisingColored(org.opencv.core.Mat,%20org.opencv.core.Mat,%20float))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float h) Modification of fastNlMeansDenoising function for colored images |
| static void | [**fastNlMeansDenoisingColored**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoisingColored(org.opencv.core.Mat,%20org.opencv.core.Mat,%20float,%20float))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float h, float hColor) Modification of fastNlMeansDenoising function for colored images |
| static void | [**fastNlMeansDenoisingColored**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoisingColored(org.opencv.core.Mat,%20org.opencv.core.Mat,%20float,%20float,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float h, float hColor, int templateWindowSize) Modification of fastNlMeansDenoising function for colored images |
| static void | [**fastNlMeansDenoisingColored**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoisingColored(org.opencv.core.Mat,%20org.opencv.core.Mat,%20float,%20float,%20int,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float h, float hColor, int templateWindowSize, int searchWindowSize) Modification of fastNlMeansDenoising function for colored images |
| static void | [**fastNlMeansDenoisingColoredMulti**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoisingColoredMulti(java.util.List,%20org.opencv.core.Mat,%20int,%20int))(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize) Modification of fastNlMeansDenoisingMulti function for colored images sequences |
| static void | [**fastNlMeansDenoisingColoredMulti**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoisingColoredMulti(java.util.List,%20org.opencv.core.Mat,%20int,%20int,%20float))(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, float h) Modification of fastNlMeansDenoisingMulti function for colored images sequences |
| static void | [**fastNlMeansDenoisingColoredMulti**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoisingColoredMulti(java.util.List,%20org.opencv.core.Mat,%20int,%20int,%20float,%20float))(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, float h, float hColor) Modification of fastNlMeansDenoisingMulti function for colored images sequences |
| static void | [**fastNlMeansDenoisingColoredMulti**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoisingColoredMulti(java.util.List,%20org.opencv.core.Mat,%20int,%20int,%20float,%20float,%20int))(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, float h, float hColor, int templateWindowSize) Modification of fastNlMeansDenoisingMulti function for colored images sequences |
| static void | [**fastNlMeansDenoisingColoredMulti**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoisingColoredMulti(java.util.List,%20org.opencv.core.Mat,%20int,%20int,%20float,%20float,%20int,%20int))(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, float h, float hColor, int templateWindowSize, int searchWindowSize) Modification of fastNlMeansDenoisingMulti function for colored images sequences |
| static void | [**fastNlMeansDenoisingMulti**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoisingMulti(java.util.List,%20org.opencv.core.Mat,%20int,%20int))(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize) Modification of fastNlMeansDenoising function for images sequence where consecutive images have been captured in small period of time. |
| static void | [**fastNlMeansDenoisingMulti**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoisingMulti(java.util.List,%20org.opencv.core.Mat,%20int,%20int,%20float))(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, float h) Modification of fastNlMeansDenoising function for images sequence where consecutive images have been captured in small period of time. |
| static void | [**fastNlMeansDenoisingMulti**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoisingMulti(java.util.List,%20org.opencv.core.Mat,%20int,%20int,%20float,%20int))(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, float h, int templateWindowSize) Modification of fastNlMeansDenoising function for images sequence where consecutive images have been captured in small period of time. |
| static void | [**fastNlMeansDenoisingMulti**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoisingMulti(java.util.List,%20org.opencv.core.Mat,%20int,%20int,%20float,%20int,%20int))(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, float h, int templateWindowSize, int searchWindowSize) Modification of fastNlMeansDenoising function for images sequence where consecutive images have been captured in small period of time. |
| static void | [**fastNlMeansDenoisingMulti**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoisingMulti(java.util.List,%20org.opencv.core.Mat,%20int,%20int,%20org.opencv.core.MatOfFloat))(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) h) Modification of fastNlMeansDenoising function for images sequence where consecutive images have been captured in small period of time. |
| static void | [**fastNlMeansDenoisingMulti**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoisingMulti(java.util.List,%20org.opencv.core.Mat,%20int,%20int,%20org.opencv.core.MatOfFloat,%20int))(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) h, int templateWindowSize) Modification of fastNlMeansDenoising function for images sequence where consecutive images have been captured in small period of time. |
| static void | [**fastNlMeansDenoisingMulti**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoisingMulti(java.util.List,%20org.opencv.core.Mat,%20int,%20int,%20org.opencv.core.MatOfFloat,%20int,%20int))(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) h, int templateWindowSize, int searchWindowSize) Modification of fastNlMeansDenoising function for images sequence where consecutive images have been captured in small period of time. |
| static void | [**fastNlMeansDenoisingMulti**](http://docs.google.com/org/opencv/photo/Photo.html#fastNlMeansDenoisingMulti(java.util.List,%20org.opencv.core.Mat,%20int,%20int,%20org.opencv.core.MatOfFloat,%20int,%20int,%20int))(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) h, int templateWindowSize, int searchWindowSize, int normType) Modification of fastNlMeansDenoising function for images sequence where consecutive images have been captured in small period of time. |
| static void | [**illuminationChange**](http://docs.google.com/org/opencv/photo/Photo.html#illuminationChange(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) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Applying an appropriate non-linear transformation to the gradient field inside the selection and then integrating back with a Poisson solver, modifies locally the apparent illumination of an image. |
| static void | [**illuminationChange**](http://docs.google.com/org/opencv/photo/Photo.html#illuminationChange(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20float))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float alpha) Applying an appropriate non-linear transformation to the gradient field inside the selection and then integrating back with a Poisson solver, modifies locally the apparent illumination of an image. |
| static void | [**illuminationChange**](http://docs.google.com/org/opencv/photo/Photo.html#illuminationChange(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20float,%20float))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float alpha, float beta) Applying an appropriate non-linear transformation to the gradient field inside the selection and then integrating back with a Poisson solver, modifies locally the apparent illumination of an image. |
| static void | [**inpaint**](http://docs.google.com/org/opencv/photo/Photo.html#inpaint(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20double,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) inpaintMask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, double inpaintRadius, int flags) Restores the selected region in an image using the region neighborhood. |
| static void | [**pencilSketch**](http://docs.google.com/org/opencv/photo/Photo.html#pencilSketch(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) dst1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst2) Pencil-like non-photorealistic line drawing |
| static void | [**pencilSketch**](http://docs.google.com/org/opencv/photo/Photo.html#pencilSketch(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20float))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst2, float sigma\_s) Pencil-like non-photorealistic line drawing |
| static void | [**pencilSketch**](http://docs.google.com/org/opencv/photo/Photo.html#pencilSketch(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20float,%20float))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst2, float sigma\_s, float sigma\_r) Pencil-like non-photorealistic line drawing |
| static void | [**pencilSketch**](http://docs.google.com/org/opencv/photo/Photo.html#pencilSketch(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20float,%20float,%20float))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst2, float sigma\_s, float sigma\_r, float shade\_factor) Pencil-like non-photorealistic line drawing |
| static void | [**seamlessClone**](http://docs.google.com/org/opencv/photo/Photo.html#seamlessClone(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Point,%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) mask, [Point](http://docs.google.com/org/opencv/core/Point.html) p, [Mat](http://docs.google.com/org/opencv/core/Mat.html) blend, int flags) Image editing tasks concern either global changes (color/intensity corrections, filters, deformations) or local changes concerned to a selection. |
| static void | [**stylization**](http://docs.google.com/org/opencv/photo/Photo.html#stylization(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) Stylization aims to produce digital imagery with a wide variety of effects not focused on photorealism. |
| static void | [**stylization**](http://docs.google.com/org/opencv/photo/Photo.html#stylization(org.opencv.core.Mat,%20org.opencv.core.Mat,%20float))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float sigma\_s) Stylization aims to produce digital imagery with a wide variety of effects not focused on photorealism. |
| static void | [**stylization**](http://docs.google.com/org/opencv/photo/Photo.html#stylization(org.opencv.core.Mat,%20org.opencv.core.Mat,%20float,%20float))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float sigma\_s, float sigma\_r) Stylization aims to produce digital imagery with a wide variety of effects not focused on photorealism. |
| static void | [**textureFlattening**](http://docs.google.com/org/opencv/photo/Photo.html#textureFlattening(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) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) By retaining only the gradients at edge locations, before integrating with the Poisson solver, one washes out the texture of the selected region, giving its contents a flat aspect. |
| static void | [**textureFlattening**](http://docs.google.com/org/opencv/photo/Photo.html#textureFlattening(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20float))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float low\_threshold) By retaining only the gradients at edge locations, before integrating with the Poisson solver, one washes out the texture of the selected region, giving its contents a flat aspect. |
| static void | [**textureFlattening**](http://docs.google.com/org/opencv/photo/Photo.html#textureFlattening(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20float,%20float))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float low\_threshold, float high\_threshold) By retaining only the gradients at edge locations, before integrating with the Poisson solver, one washes out the texture of the selected region, giving its contents a flat aspect. |
| static void | [**textureFlattening**](http://docs.google.com/org/opencv/photo/Photo.html#textureFlattening(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20float,%20float,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float low\_threshold, float high\_threshold, int kernel\_size) By retaining only the gradients at edge locations, before integrating with the Poisson solver, one washes out the texture of the selected region, giving its contents a flat aspect. |

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

### Field Detail

#### INPAINT\_NS public static final int INPAINT\_NSSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.photo.Photo.INPAINT_NS)

#### INPAINT\_TELEA public static final int INPAINT\_TELEASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.photo.Photo.INPAINT_TELEA)

#### LDR\_SIZE public static final int LDR\_SIZESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.photo.Photo.LDR_SIZE)

#### MIXED\_CLONE public static final int MIXED\_CLONESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.photo.Photo.MIXED_CLONE)

#### MONOCHROME\_TRANSFER public static final int MONOCHROME\_TRANSFERSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.photo.Photo.MONOCHROME_TRANSFER)

#### NORMAL\_CLONE public static final int NORMAL\_CLONESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.photo.Photo.NORMAL_CLONE)

#### NORMCONV\_FILTER public static final int NORMCONV\_FILTERSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.photo.Photo.NORMCONV_FILTER)

#### RECURS\_FILTER public static final int RECURS\_FILTERSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.photo.Photo.RECURS_FILTER)

### Constructor Detail

#### Photo public Photo()

### Method Detail

#### colorChange public static void colorChange([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Given an original color image, two differently colored versions of this image can be mixed seamlessly.Parameters:src - Input 8-bit 3-channel image.mask - Input 8-bit 1 or 3-channel image.dst - Output image with the same size and type as src . Multiplication factor is between .5 to 2.5.

#### colorChange public static void colorChange([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float red\_mul) Given an original color image, two differently colored versions of this image can be mixed seamlessly.Parameters:src - Input 8-bit 3-channel image.mask - Input 8-bit 1 or 3-channel image.dst - Output image with the same size and type as src .red\_mul - R-channel multiply factor. Multiplication factor is between .5 to 2.5.

#### colorChange public static void colorChange([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float red\_mul, float green\_mul) Given an original color image, two differently colored versions of this image can be mixed seamlessly.Parameters:src - Input 8-bit 3-channel image.mask - Input 8-bit 1 or 3-channel image.dst - Output image with the same size and type as src .red\_mul - R-channel multiply factor.green\_mul - G-channel multiply factor. Multiplication factor is between .5 to 2.5.

#### colorChange public static void colorChange([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float red\_mul, float green\_mul, float blue\_mul) Given an original color image, two differently colored versions of this image can be mixed seamlessly.Parameters:src - Input 8-bit 3-channel image.mask - Input 8-bit 1 or 3-channel image.dst - Output image with the same size and type as src .red\_mul - R-channel multiply factor.green\_mul - G-channel multiply factor.blue\_mul - B-channel multiply factor. Multiplication factor is between .5 to 2.5.

#### createAlignMTB public static [AlignMTB](http://docs.google.com/org/opencv/photo/AlignMTB.html) createAlignMTB() Creates AlignMTB object usually good enough (31 and 63 pixels shift respectively). median value.Returns:automatically generated

#### createAlignMTB public static [AlignMTB](http://docs.google.com/org/opencv/photo/AlignMTB.html) createAlignMTB(int max\_bits) Creates AlignMTB objectParameters:max\_bits - logarithm to the base 2 of maximal shift in each dimension. Values of 5 and 6 are usually good enough (31 and 63 pixels shift respectively). median value. Returns:automatically generated

#### createAlignMTB public static [AlignMTB](http://docs.google.com/org/opencv/photo/AlignMTB.html) createAlignMTB(int max\_bits, int exclude\_range) Creates AlignMTB objectParameters:max\_bits - logarithm to the base 2 of maximal shift in each dimension. Values of 5 and 6 are usually good enough (31 and 63 pixels shift respectively).exclude\_range - range for exclusion bitmap that is constructed to suppress noise around the median value. Returns:automatically generated

#### createAlignMTB public static [AlignMTB](http://docs.google.com/org/opencv/photo/AlignMTB.html) createAlignMTB(int max\_bits, int exclude\_range, boolean cut) Creates AlignMTB objectParameters:max\_bits - logarithm to the base 2 of maximal shift in each dimension. Values of 5 and 6 are usually good enough (31 and 63 pixels shift respectively).exclude\_range - range for exclusion bitmap that is constructed to suppress noise around the median value.cut - if true cuts images, otherwise fills the new regions with zeros. Returns:automatically generated

#### createCalibrateDebevec public static [CalibrateDebevec](http://docs.google.com/org/opencv/photo/CalibrateDebevec.html) createCalibrateDebevec() Creates CalibrateDebevec object response. rectangular grid.Returns:automatically generated

#### createCalibrateDebevec public static [CalibrateDebevec](http://docs.google.com/org/opencv/photo/CalibrateDebevec.html) createCalibrateDebevec(int samples) Creates CalibrateDebevec objectParameters:samples - number of pixel locations to use response. rectangular grid. Returns:automatically generated

#### createCalibrateDebevec public static [CalibrateDebevec](http://docs.google.com/org/opencv/photo/CalibrateDebevec.html) createCalibrateDebevec(int samples, float lambda) Creates CalibrateDebevec objectParameters:samples - number of pixel locations to uselambda - smoothness term weight. Greater values produce smoother results, but can alter the response. rectangular grid. Returns:automatically generated

#### createCalibrateDebevec public static [CalibrateDebevec](http://docs.google.com/org/opencv/photo/CalibrateDebevec.html) createCalibrateDebevec(int samples, float lambda, boolean random) Creates CalibrateDebevec objectParameters:samples - number of pixel locations to uselambda - smoothness term weight. Greater values produce smoother results, but can alter the response.random - if true sample pixel locations are chosen at random, otherwise they form a rectangular grid. Returns:automatically generated

#### createCalibrateRobertson public static [CalibrateRobertson](http://docs.google.com/org/opencv/photo/CalibrateRobertson.html) createCalibrateRobertson() Creates CalibrateRobertson objectReturns:automatically generated

#### createCalibrateRobertson public static [CalibrateRobertson](http://docs.google.com/org/opencv/photo/CalibrateRobertson.html) createCalibrateRobertson(int max\_iter) Creates CalibrateRobertson objectParameters:max\_iter - maximal number of Gauss-Seidel solver iterations. Returns:automatically generated

#### createCalibrateRobertson public static [CalibrateRobertson](http://docs.google.com/org/opencv/photo/CalibrateRobertson.html) createCalibrateRobertson(int max\_iter, float threshold) Creates CalibrateRobertson objectParameters:max\_iter - maximal number of Gauss-Seidel solver iterations.threshold - target difference between results of two successive steps of the minimization. Returns:automatically generated

#### createMergeDebevec public static [MergeDebevec](http://docs.google.com/org/opencv/photo/MergeDebevec.html) createMergeDebevec() Creates MergeDebevec objectReturns:automatically generated

#### createMergeMertens public static [MergeMertens](http://docs.google.com/org/opencv/photo/MergeMertens.html) createMergeMertens() Creates MergeMertens objectReturns:automatically generated

#### createMergeMertens public static [MergeMertens](http://docs.google.com/org/opencv/photo/MergeMertens.html) createMergeMertens(float contrast\_weight) Creates MergeMertens objectParameters:contrast\_weight - contrast measure weight. See MergeMertens. Returns:automatically generated

#### createMergeMertens public static [MergeMertens](http://docs.google.com/org/opencv/photo/MergeMertens.html) createMergeMertens(float contrast\_weight, float saturation\_weight) Creates MergeMertens objectParameters:contrast\_weight - contrast measure weight. See MergeMertens.saturation\_weight - saturation measure weight Returns:automatically generated

#### createMergeMertens public static [MergeMertens](http://docs.google.com/org/opencv/photo/MergeMertens.html) createMergeMertens(float contrast\_weight, float saturation\_weight, float exposure\_weight) Creates MergeMertens objectParameters:contrast\_weight - contrast measure weight. See MergeMertens.saturation\_weight - saturation measure weightexposure\_weight - well-exposedness measure weight Returns:automatically generated

#### createMergeRobertson public static [MergeRobertson](http://docs.google.com/org/opencv/photo/MergeRobertson.html) createMergeRobertson() Creates MergeRobertson objectReturns:automatically generated

#### createTonemap public static [Tonemap](http://docs.google.com/org/opencv/photo/Tonemap.html) createTonemap() Creates simple linear mapper with gamma correction equal to 2.2f is suitable for most displays. Generally gamma > 1 brightens the image and gamma < 1 darkens it.Returns:automatically generated

#### createTonemap public static [Tonemap](http://docs.google.com/org/opencv/photo/Tonemap.html) createTonemap(float gamma) Creates simple linear mapper with gamma correctionParameters:gamma - positive value for gamma correction. Gamma value of 1.0 implies no correction, gamma equal to 2.2f is suitable for most displays. Generally gamma > 1 brightens the image and gamma < 1 darkens it. Returns:automatically generated

#### createTonemapDrago public static [TonemapDrago](http://docs.google.com/org/opencv/photo/TonemapDrago.html) createTonemapDrago() Creates TonemapDrago object than 1 increase saturation and values less than 1 decrease it. results, default value is 0.85.Returns:automatically generated

#### createTonemapDrago public static [TonemapDrago](http://docs.google.com/org/opencv/photo/TonemapDrago.html) createTonemapDrago(float gamma) Creates TonemapDrago objectParameters:gamma - gamma value for gamma correction. See createTonemap than 1 increase saturation and values less than 1 decrease it. results, default value is 0.85. Returns:automatically generated

#### createTonemapDrago public static [TonemapDrago](http://docs.google.com/org/opencv/photo/TonemapDrago.html) createTonemapDrago(float gamma, float saturation) Creates TonemapDrago objectParameters:gamma - gamma value for gamma correction. See createTonemapsaturation - positive saturation enhancement value. 1.0 preserves saturation, values greater than 1 increase saturation and values less than 1 decrease it. results, default value is 0.85. Returns:automatically generated

#### createTonemapDrago public static [TonemapDrago](http://docs.google.com/org/opencv/photo/TonemapDrago.html) createTonemapDrago(float gamma, float saturation, float bias) Creates TonemapDrago objectParameters:gamma - gamma value for gamma correction. See createTonemapsaturation - positive saturation enhancement value. 1.0 preserves saturation, values greater than 1 increase saturation and values less than 1 decrease it.bias - value for bias function in [0, 1] range. Values from 0.7 to 0.9 usually give best results, default value is 0.85. Returns:automatically generated

#### createTonemapMantiuk public static [TonemapMantiuk](http://docs.google.com/org/opencv/photo/TonemapMantiuk.html) createTonemapMantiuk() Creates TonemapMantiuk object dynamic range. Values from 0.6 to 0.9 produce best results.Returns:automatically generated

#### createTonemapMantiuk public static [TonemapMantiuk](http://docs.google.com/org/opencv/photo/TonemapMantiuk.html) createTonemapMantiuk(float gamma) Creates TonemapMantiuk objectParameters:gamma - gamma value for gamma correction. See createTonemap dynamic range. Values from 0.6 to 0.9 produce best results. Returns:automatically generated

#### createTonemapMantiuk public static [TonemapMantiuk](http://docs.google.com/org/opencv/photo/TonemapMantiuk.html) createTonemapMantiuk(float gamma, float scale) Creates TonemapMantiuk objectParameters:gamma - gamma value for gamma correction. See createTonemapscale - contrast scale factor. HVS response is multiplied by this parameter, thus compressing dynamic range. Values from 0.6 to 0.9 produce best results. Returns:automatically generated

#### createTonemapMantiuk public static [TonemapMantiuk](http://docs.google.com/org/opencv/photo/TonemapMantiuk.html) createTonemapMantiuk(float gamma, float scale, float saturation) Creates TonemapMantiuk objectParameters:gamma - gamma value for gamma correction. See createTonemapscale - contrast scale factor. HVS response is multiplied by this parameter, thus compressing dynamic range. Values from 0.6 to 0.9 produce best results.saturation - saturation enhancement value. See createTonemapDrago Returns:automatically generated

#### createTonemapReinhard public static [TonemapReinhard](http://docs.google.com/org/opencv/photo/TonemapReinhard.html) createTonemapReinhard() Creates TonemapReinhard object value, if 0 it's global, otherwise it's a weighted mean of this two cases. if 0 adaptation level is the same for each channel.Returns:automatically generated

#### createTonemapReinhard public static [TonemapReinhard](http://docs.google.com/org/opencv/photo/TonemapReinhard.html) createTonemapReinhard(float gamma) Creates TonemapReinhard objectParameters:gamma - gamma value for gamma correction. See createTonemap value, if 0 it's global, otherwise it's a weighted mean of this two cases. if 0 adaptation level is the same for each channel. Returns:automatically generated

#### createTonemapReinhard public static [TonemapReinhard](http://docs.google.com/org/opencv/photo/TonemapReinhard.html) createTonemapReinhard(float gamma, float intensity) Creates TonemapReinhard objectParameters:gamma - gamma value for gamma correction. See createTonemapintensity - result intensity in [-8, 8] range. Greater intensity produces brighter results. value, if 0 it's global, otherwise it's a weighted mean of this two cases. if 0 adaptation level is the same for each channel. Returns:automatically generated

#### createTonemapReinhard public static [TonemapReinhard](http://docs.google.com/org/opencv/photo/TonemapReinhard.html) createTonemapReinhard(float gamma, float intensity, float light\_adapt) Creates TonemapReinhard objectParameters:gamma - gamma value for gamma correction. See createTonemapintensity - result intensity in [-8, 8] range. Greater intensity produces brighter results.light\_adapt - light adaptation in [0, 1] range. If 1 adaptation is based only on pixel value, if 0 it's global, otherwise it's a weighted mean of this two cases. if 0 adaptation level is the same for each channel. Returns:automatically generated

#### createTonemapReinhard public static [TonemapReinhard](http://docs.google.com/org/opencv/photo/TonemapReinhard.html) createTonemapReinhard(float gamma, float intensity, float light\_adapt, float color\_adapt) Creates TonemapReinhard objectParameters:gamma - gamma value for gamma correction. See createTonemapintensity - result intensity in [-8, 8] range. Greater intensity produces brighter results.light\_adapt - light adaptation in [0, 1] range. If 1 adaptation is based only on pixel value, if 0 it's global, otherwise it's a weighted mean of this two cases.color\_adapt - chromatic adaptation in [0, 1] range. If 1 channels are treated independently, if 0 adaptation level is the same for each channel. Returns:automatically generated

#### decolor public static void decolor([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) grayscale, [Mat](http://docs.google.com/org/opencv/core/Mat.html) color\_boost) Transforms a color image to a grayscale image. It is a basic tool in digital printing, stylized black-and-white photograph rendering, and in many single channel image processing applications CITE: CL12 .Parameters:src - Input 8-bit 3-channel image.grayscale - Output 8-bit 1-channel image.color\_boost - Output 8-bit 3-channel image. This function is to be applied on color images.

#### denoise\_TVL1 public static void denoise\_TVL1(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> observations, [Mat](http://docs.google.com/org/opencv/core/Mat.html) result) Primal-dual algorithm is an algorithm for solving special types of variational problems (that is, finding a function to minimize some functional). As the image denoising, in particular, may be seen as the variational problem, primal-dual algorithm then can be used to perform denoising and this is exactly what is implemented. It should be noted, that this implementation was taken from the July 2013 blog entry CITE: MA13 , which also contained (slightly more general) ready-to-use source code on Python. Subsequently, that code was rewritten on C++ with the usage of openCV by Vadim Pisarevsky at the end of July 2013 and finally it was slightly adapted by later authors. Although the thorough discussion and justification of the algorithm involved may be found in CITE: ChambolleEtAl, it might make sense to skim over it here, following CITE: MA13 . To begin with, we consider the 1-byte gray-level images as the functions from the rectangular domain of pixels (it may be seen as set \(\left\{(x,y)\in\mathbb{N}\times\mathbb{N}\mid 1\leq x\leq n,\;1\leq y\leq m\right\}\) for some \(m,\;n\in\mathbb{N}\)) into \(\{0,1,\dots,255\}\). We shall denote the noised images as \(f\_i\) and with this view, given some image \(x\) of the same size, we may measure how bad it is by the formula \(\left\|\left\|\nabla x\right\|\right\| + \lambda\sum\_i\left\|\left\|x-f\_i\right\|\right\|\) \(\|\|\cdot\|\|\) here denotes \(L\_2\)-norm and as you see, the first addend states that we want our image to be smooth (ideally, having zero gradient, thus being constant) and the second states that we want our result to be close to the observations we've got. If we treat \(x\) as a function, this is exactly the functional what we seek to minimize and here the Primal-Dual algorithm comes into play.Parameters:observations - This array should contain one or more noised versions of the image that is to be restored.result - Here the denoised image will be stored. There is no need to do pre-allocation of storage space, as it will be automatically allocated, if necessary. (blurred) images are treated more favorably than detailed (but maybe more noised) ones. Roughly speaking, as it becomes smaller, the result will be more blur but more sever outliers will be removed. better, but it is hard to quantitatively refine this statement, so just use the default and increase it if the results are poor.

#### denoise\_TVL1 public static void denoise\_TVL1(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> observations, [Mat](http://docs.google.com/org/opencv/core/Mat.html) result, double lambda) Primal-dual algorithm is an algorithm for solving special types of variational problems (that is, finding a function to minimize some functional). As the image denoising, in particular, may be seen as the variational problem, primal-dual algorithm then can be used to perform denoising and this is exactly what is implemented. It should be noted, that this implementation was taken from the July 2013 blog entry CITE: MA13 , which also contained (slightly more general) ready-to-use source code on Python. Subsequently, that code was rewritten on C++ with the usage of openCV by Vadim Pisarevsky at the end of July 2013 and finally it was slightly adapted by later authors. Although the thorough discussion and justification of the algorithm involved may be found in CITE: ChambolleEtAl, it might make sense to skim over it here, following CITE: MA13 . To begin with, we consider the 1-byte gray-level images as the functions from the rectangular domain of pixels (it may be seen as set \(\left\{(x,y)\in\mathbb{N}\times\mathbb{N}\mid 1\leq x\leq n,\;1\leq y\leq m\right\}\) for some \(m,\;n\in\mathbb{N}\)) into \(\{0,1,\dots,255\}\). We shall denote the noised images as \(f\_i\) and with this view, given some image \(x\) of the same size, we may measure how bad it is by the formula \(\left\|\left\|\nabla x\right\|\right\| + \lambda\sum\_i\left\|\left\|x-f\_i\right\|\right\|\) \(\|\|\cdot\|\|\) here denotes \(L\_2\)-norm and as you see, the first addend states that we want our image to be smooth (ideally, having zero gradient, thus being constant) and the second states that we want our result to be close to the observations we've got. If we treat \(x\) as a function, this is exactly the functional what we seek to minimize and here the Primal-Dual algorithm comes into play.Parameters:observations - This array should contain one or more noised versions of the image that is to be restored.result - Here the denoised image will be stored. There is no need to do pre-allocation of storage space, as it will be automatically allocated, if necessary.lambda - Corresponds to \(\lambda\) in the formulas above. As it is enlarged, the smooth (blurred) images are treated more favorably than detailed (but maybe more noised) ones. Roughly speaking, as it becomes smaller, the result will be more blur but more sever outliers will be removed. better, but it is hard to quantitatively refine this statement, so just use the default and increase it if the results are poor.

#### denoise\_TVL1 public static void denoise\_TVL1(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> observations, [Mat](http://docs.google.com/org/opencv/core/Mat.html) result, double lambda, int niters) Primal-dual algorithm is an algorithm for solving special types of variational problems (that is, finding a function to minimize some functional). As the image denoising, in particular, may be seen as the variational problem, primal-dual algorithm then can be used to perform denoising and this is exactly what is implemented. It should be noted, that this implementation was taken from the July 2013 blog entry CITE: MA13 , which also contained (slightly more general) ready-to-use source code on Python. Subsequently, that code was rewritten on C++ with the usage of openCV by Vadim Pisarevsky at the end of July 2013 and finally it was slightly adapted by later authors. Although the thorough discussion and justification of the algorithm involved may be found in CITE: ChambolleEtAl, it might make sense to skim over it here, following CITE: MA13 . To begin with, we consider the 1-byte gray-level images as the functions from the rectangular domain of pixels (it may be seen as set \(\left\{(x,y)\in\mathbb{N}\times\mathbb{N}\mid 1\leq x\leq n,\;1\leq y\leq m\right\}\) for some \(m,\;n\in\mathbb{N}\)) into \(\{0,1,\dots,255\}\). We shall denote the noised images as \(f\_i\) and with this view, given some image \(x\) of the same size, we may measure how bad it is by the formula \(\left\|\left\|\nabla x\right\|\right\| + \lambda\sum\_i\left\|\left\|x-f\_i\right\|\right\|\) \(\|\|\cdot\|\|\) here denotes \(L\_2\)-norm and as you see, the first addend states that we want our image to be smooth (ideally, having zero gradient, thus being constant) and the second states that we want our result to be close to the observations we've got. If we treat \(x\) as a function, this is exactly the functional what we seek to minimize and here the Primal-Dual algorithm comes into play.Parameters:observations - This array should contain one or more noised versions of the image that is to be restored.result - Here the denoised image will be stored. There is no need to do pre-allocation of storage space, as it will be automatically allocated, if necessary.lambda - Corresponds to \(\lambda\) in the formulas above. As it is enlarged, the smooth (blurred) images are treated more favorably than detailed (but maybe more noised) ones. Roughly speaking, as it becomes smaller, the result will be more blur but more sever outliers will be removed.niters - Number of iterations that the algorithm will run. Of course, as more iterations as better, but it is hard to quantitatively refine this statement, so just use the default and increase it if the results are poor.

#### detailEnhance public static void detailEnhance([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) This filter enhances the details of a particular image.Parameters:src - Input 8-bit 3-channel image.dst - Output image with the same size and type as src.

#### detailEnhance public static void detailEnhance([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float sigma\_s) This filter enhances the details of a particular image.Parameters:src - Input 8-bit 3-channel image.dst - Output image with the same size and type as src.sigma\_s - %Range between 0 to 200.

#### detailEnhance public static void detailEnhance([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float sigma\_s, float sigma\_r) This filter enhances the details of a particular image.Parameters:src - Input 8-bit 3-channel image.dst - Output image with the same size and type as src.sigma\_s - %Range between 0 to 200.sigma\_r - %Range between 0 to 1.

#### edgePreservingFilter public static void edgePreservingFilter([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Filtering is the fundamental operation in image and video processing. Edge-preserving smoothing filters are used in many different applications CITE: EM11 .Parameters:src - Input 8-bit 3-channel image.dst - Output 8-bit 3-channel image.

#### edgePreservingFilter public static void edgePreservingFilter([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int flags) Filtering is the fundamental operation in image and video processing. Edge-preserving smoothing filters are used in many different applications CITE: EM11 .Parameters:src - Input 8-bit 3-channel image.dst - Output 8-bit 3-channel image.flags - Edge preserving filters: cv::RECURS\_FILTER or cv::NORMCONV\_FILTER

#### edgePreservingFilter public static void edgePreservingFilter([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int flags, float sigma\_s) Filtering is the fundamental operation in image and video processing. Edge-preserving smoothing filters are used in many different applications CITE: EM11 .Parameters:src - Input 8-bit 3-channel image.dst - Output 8-bit 3-channel image.flags - Edge preserving filters: cv::RECURS\_FILTER or cv::NORMCONV\_FILTERsigma\_s - %Range between 0 to 200.

#### edgePreservingFilter public static void edgePreservingFilter([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int flags, float sigma\_s, float sigma\_r) Filtering is the fundamental operation in image and video processing. Edge-preserving smoothing filters are used in many different applications CITE: EM11 .Parameters:src - Input 8-bit 3-channel image.dst - Output 8-bit 3-channel image.flags - Edge preserving filters: cv::RECURS\_FILTER or cv::NORMCONV\_FILTERsigma\_s - %Range between 0 to 200.sigma\_r - %Range between 0 to 1.

#### fastNlMeansDenoising public static void fastNlMeansDenoising([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Perform image denoising using Non-local Means Denoising algorithm <http://www.ipol.im/pub/algo/bcm\_non\_local\_means\_denoising/> with several computational optimizations. Noise expected to be a gaussian white noiseParameters:src - Input 8-bit 1-channel, 2-channel, 3-channel or 4-channel image.dst - Output image with the same size and type as src . Should be odd. Recommended value 7 pixels given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixels removes image details, smaller h value preserves details but also preserves some noise This function expected to be applied to grayscale images. For colored images look at fastNlMeansDenoisingColored. Advanced usage of this functions can be manual denoising of colored image in different colorspaces. Such approach is used in fastNlMeansDenoisingColored by converting image to CIELAB colorspace and then separately denoise L and AB components with different h parameter.

#### fastNlMeansDenoising public static void fastNlMeansDenoising([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float h) Perform image denoising using Non-local Means Denoising algorithm <http://www.ipol.im/pub/algo/bcm\_non\_local\_means\_denoising/> with several computational optimizations. Noise expected to be a gaussian white noiseParameters:src - Input 8-bit 1-channel, 2-channel, 3-channel or 4-channel image.dst - Output image with the same size and type as src . Should be odd. Recommended value 7 pixels given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Parameter regulating filter strength. Big h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noise This function expected to be applied to grayscale images. For colored images look at fastNlMeansDenoisingColored. Advanced usage of this functions can be manual denoising of colored image in different colorspaces. Such approach is used in fastNlMeansDenoisingColored by converting image to CIELAB colorspace and then separately denoise L and AB components with different h parameter.

#### fastNlMeansDenoising public static void fastNlMeansDenoising([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float h, int templateWindowSize) Perform image denoising using Non-local Means Denoising algorithm <http://www.ipol.im/pub/algo/bcm\_non\_local\_means\_denoising/> with several computational optimizations. Noise expected to be a gaussian white noiseParameters:src - Input 8-bit 1-channel, 2-channel, 3-channel or 4-channel image.dst - Output image with the same size and type as src .templateWindowSize - Size in pixels of the template patch that is used to compute weights. Should be odd. Recommended value 7 pixels given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Parameter regulating filter strength. Big h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noise This function expected to be applied to grayscale images. For colored images look at fastNlMeansDenoisingColored. Advanced usage of this functions can be manual denoising of colored image in different colorspaces. Such approach is used in fastNlMeansDenoisingColored by converting image to CIELAB colorspace and then separately denoise L and AB components with different h parameter.

#### fastNlMeansDenoising public static void fastNlMeansDenoising([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float h, int templateWindowSize, int searchWindowSize) Perform image denoising using Non-local Means Denoising algorithm <http://www.ipol.im/pub/algo/bcm\_non\_local\_means\_denoising/> with several computational optimizations. Noise expected to be a gaussian white noiseParameters:src - Input 8-bit 1-channel, 2-channel, 3-channel or 4-channel image.dst - Output image with the same size and type as src .templateWindowSize - Size in pixels of the template patch that is used to compute weights. Should be odd. Recommended value 7 pixelssearchWindowSize - Size in pixels of the window that is used to compute weighted average for given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Parameter regulating filter strength. Big h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noise This function expected to be applied to grayscale images. For colored images look at fastNlMeansDenoisingColored. Advanced usage of this functions can be manual denoising of colored image in different colorspaces. Such approach is used in fastNlMeansDenoisingColored by converting image to CIELAB colorspace and then separately denoise L and AB components with different h parameter.

#### fastNlMeansDenoising public static void fastNlMeansDenoising([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) h) Perform image denoising using Non-local Means Denoising algorithm <http://www.ipol.im/pub/algo/bcm\_non\_local\_means\_denoising/> with several computational optimizations. Noise expected to be a gaussian white noiseParameters:src - Input 8-bit or 16-bit (only with NORM\_L1) 1-channel, 2-channel, 3-channel or 4-channel image.dst - Output image with the same size and type as src . Should be odd. Recommended value 7 pixels given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Array of parameters regulating filter strength, either one parameter applied to all channels or one per channel in dst. Big h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noise This function expected to be applied to grayscale images. For colored images look at fastNlMeansDenoisingColored. Advanced usage of this functions can be manual denoising of colored image in different colorspaces. Such approach is used in fastNlMeansDenoisingColored by converting image to CIELAB colorspace and then separately denoise L and AB components with different h parameter.

#### fastNlMeansDenoising public static void fastNlMeansDenoising([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) h, int templateWindowSize) Perform image denoising using Non-local Means Denoising algorithm <http://www.ipol.im/pub/algo/bcm\_non\_local\_means\_denoising/> with several computational optimizations. Noise expected to be a gaussian white noiseParameters:src - Input 8-bit or 16-bit (only with NORM\_L1) 1-channel, 2-channel, 3-channel or 4-channel image.dst - Output image with the same size and type as src .templateWindowSize - Size in pixels of the template patch that is used to compute weights. Should be odd. Recommended value 7 pixels given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Array of parameters regulating filter strength, either one parameter applied to all channels or one per channel in dst. Big h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noise This function expected to be applied to grayscale images. For colored images look at fastNlMeansDenoisingColored. Advanced usage of this functions can be manual denoising of colored image in different colorspaces. Such approach is used in fastNlMeansDenoisingColored by converting image to CIELAB colorspace and then separately denoise L and AB components with different h parameter.

#### fastNlMeansDenoising public static void fastNlMeansDenoising([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) h, int templateWindowSize, int searchWindowSize) Perform image denoising using Non-local Means Denoising algorithm <http://www.ipol.im/pub/algo/bcm\_non\_local\_means\_denoising/> with several computational optimizations. Noise expected to be a gaussian white noiseParameters:src - Input 8-bit or 16-bit (only with NORM\_L1) 1-channel, 2-channel, 3-channel or 4-channel image.dst - Output image with the same size and type as src .templateWindowSize - Size in pixels of the template patch that is used to compute weights. Should be odd. Recommended value 7 pixelssearchWindowSize - Size in pixels of the window that is used to compute weighted average for given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Array of parameters regulating filter strength, either one parameter applied to all channels or one per channel in dst. Big h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noise This function expected to be applied to grayscale images. For colored images look at fastNlMeansDenoisingColored. Advanced usage of this functions can be manual denoising of colored image in different colorspaces. Such approach is used in fastNlMeansDenoisingColored by converting image to CIELAB colorspace and then separately denoise L and AB components with different h parameter.

#### fastNlMeansDenoising public static void fastNlMeansDenoising([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) h, int templateWindowSize, int searchWindowSize, int normType) Perform image denoising using Non-local Means Denoising algorithm <http://www.ipol.im/pub/algo/bcm\_non\_local\_means\_denoising/> with several computational optimizations. Noise expected to be a gaussian white noiseParameters:src - Input 8-bit or 16-bit (only with NORM\_L1) 1-channel, 2-channel, 3-channel or 4-channel image.dst - Output image with the same size and type as src .templateWindowSize - Size in pixels of the template patch that is used to compute weights. Should be odd. Recommended value 7 pixelssearchWindowSize - Size in pixels of the window that is used to compute weighted average for given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Array of parameters regulating filter strength, either one parameter applied to all channels or one per channel in dst. Big h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noisenormType - Type of norm used for weight calculation. Can be either NORM\_L2 or NORM\_L1 This function expected to be applied to grayscale images. For colored images look at fastNlMeansDenoisingColored. Advanced usage of this functions can be manual denoising of colored image in different colorspaces. Such approach is used in fastNlMeansDenoisingColored by converting image to CIELAB colorspace and then separately denoise L and AB components with different h parameter.

#### fastNlMeansDenoisingColored public static void fastNlMeansDenoisingColored([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Modification of fastNlMeansDenoising function for colored imagesParameters:src - Input 8-bit 3-channel image.dst - Output image with the same size and type as src . Should be odd. Recommended value 7 pixels given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixels removes noise but also removes image details, smaller h value preserves details but also preserves some noise will be enough to remove colored noise and do not distort colors The function converts image to CIELAB colorspace and then separately denoise L and AB components with given h parameters using fastNlMeansDenoising function.

#### fastNlMeansDenoisingColored public static void fastNlMeansDenoisingColored([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float h) Modification of fastNlMeansDenoising function for colored imagesParameters:src - Input 8-bit 3-channel image.dst - Output image with the same size and type as src . Should be odd. Recommended value 7 pixels given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Parameter regulating filter strength for luminance component. Bigger h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noise will be enough to remove colored noise and do not distort colors The function converts image to CIELAB colorspace and then separately denoise L and AB components with given h parameters using fastNlMeansDenoising function.

#### fastNlMeansDenoisingColored public static void fastNlMeansDenoisingColored([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float h, float hColor) Modification of fastNlMeansDenoising function for colored imagesParameters:src - Input 8-bit 3-channel image.dst - Output image with the same size and type as src . Should be odd. Recommended value 7 pixels given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Parameter regulating filter strength for luminance component. Bigger h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noisehColor - The same as h but for color components. For most images value equals 10 will be enough to remove colored noise and do not distort colors The function converts image to CIELAB colorspace and then separately denoise L and AB components with given h parameters using fastNlMeansDenoising function.

#### fastNlMeansDenoisingColored public static void fastNlMeansDenoisingColored([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float h, float hColor, int templateWindowSize) Modification of fastNlMeansDenoising function for colored imagesParameters:src - Input 8-bit 3-channel image.dst - Output image with the same size and type as src .templateWindowSize - Size in pixels of the template patch that is used to compute weights. Should be odd. Recommended value 7 pixels given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Parameter regulating filter strength for luminance component. Bigger h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noisehColor - The same as h but for color components. For most images value equals 10 will be enough to remove colored noise and do not distort colors The function converts image to CIELAB colorspace and then separately denoise L and AB components with given h parameters using fastNlMeansDenoising function.

#### fastNlMeansDenoisingColored public static void fastNlMeansDenoisingColored([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float h, float hColor, int templateWindowSize, int searchWindowSize) Modification of fastNlMeansDenoising function for colored imagesParameters:src - Input 8-bit 3-channel image.dst - Output image with the same size and type as src .templateWindowSize - Size in pixels of the template patch that is used to compute weights. Should be odd. Recommended value 7 pixelssearchWindowSize - Size in pixels of the window that is used to compute weighted average for given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Parameter regulating filter strength for luminance component. Bigger h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noisehColor - The same as h but for color components. For most images value equals 10 will be enough to remove colored noise and do not distort colors The function converts image to CIELAB colorspace and then separately denoise L and AB components with given h parameters using fastNlMeansDenoising function.

#### fastNlMeansDenoisingColoredMulti public static void fastNlMeansDenoisingColoredMulti(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize) Modification of fastNlMeansDenoisingMulti function for colored images sequencesParameters:srcImgs - Input 8-bit 3-channel images sequence. All images should have the same type and size.imgToDenoiseIndex - Target image to denoise index in srcImgs sequencetemporalWindowSize - Number of surrounding images to use for target image denoising. Should be odd. Images from imgToDenoiseIndex - temporalWindowSize / 2 to imgToDenoiseIndex - temporalWindowSize / 2 from srcImgs will be used to denoise srcImgs[imgToDenoiseIndex] image.dst - Output image with the same size and type as srcImgs images. Should be odd. Recommended value 7 pixels given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixels removes noise but also removes image details, smaller h value preserves details but also preserves some noise. The function converts images to CIELAB colorspace and then separately denoise L and AB components with given h parameters using fastNlMeansDenoisingMulti function.

#### fastNlMeansDenoisingColoredMulti public static void fastNlMeansDenoisingColoredMulti(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, float h) Modification of fastNlMeansDenoisingMulti function for colored images sequencesParameters:srcImgs - Input 8-bit 3-channel images sequence. All images should have the same type and size.imgToDenoiseIndex - Target image to denoise index in srcImgs sequencetemporalWindowSize - Number of surrounding images to use for target image denoising. Should be odd. Images from imgToDenoiseIndex - temporalWindowSize / 2 to imgToDenoiseIndex - temporalWindowSize / 2 from srcImgs will be used to denoise srcImgs[imgToDenoiseIndex] image.dst - Output image with the same size and type as srcImgs images. Should be odd. Recommended value 7 pixels given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Parameter regulating filter strength for luminance component. Bigger h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noise. The function converts images to CIELAB colorspace and then separately denoise L and AB components with given h parameters using fastNlMeansDenoisingMulti function.

#### fastNlMeansDenoisingColoredMulti public static void fastNlMeansDenoisingColoredMulti(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, float h, float hColor) Modification of fastNlMeansDenoisingMulti function for colored images sequencesParameters:srcImgs - Input 8-bit 3-channel images sequence. All images should have the same type and size.imgToDenoiseIndex - Target image to denoise index in srcImgs sequencetemporalWindowSize - Number of surrounding images to use for target image denoising. Should be odd. Images from imgToDenoiseIndex - temporalWindowSize / 2 to imgToDenoiseIndex - temporalWindowSize / 2 from srcImgs will be used to denoise srcImgs[imgToDenoiseIndex] image.dst - Output image with the same size and type as srcImgs images. Should be odd. Recommended value 7 pixels given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Parameter regulating filter strength for luminance component. Bigger h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noise.hColor - The same as h but for color components. The function converts images to CIELAB colorspace and then separately denoise L and AB components with given h parameters using fastNlMeansDenoisingMulti function.

#### fastNlMeansDenoisingColoredMulti public static void fastNlMeansDenoisingColoredMulti(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, float h, float hColor, int templateWindowSize) Modification of fastNlMeansDenoisingMulti function for colored images sequencesParameters:srcImgs - Input 8-bit 3-channel images sequence. All images should have the same type and size.imgToDenoiseIndex - Target image to denoise index in srcImgs sequencetemporalWindowSize - Number of surrounding images to use for target image denoising. Should be odd. Images from imgToDenoiseIndex - temporalWindowSize / 2 to imgToDenoiseIndex - temporalWindowSize / 2 from srcImgs will be used to denoise srcImgs[imgToDenoiseIndex] image.dst - Output image with the same size and type as srcImgs images.templateWindowSize - Size in pixels of the template patch that is used to compute weights. Should be odd. Recommended value 7 pixels given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Parameter regulating filter strength for luminance component. Bigger h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noise.hColor - The same as h but for color components. The function converts images to CIELAB colorspace and then separately denoise L and AB components with given h parameters using fastNlMeansDenoisingMulti function.

#### fastNlMeansDenoisingColoredMulti public static void fastNlMeansDenoisingColoredMulti(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, float h, float hColor, int templateWindowSize, int searchWindowSize) Modification of fastNlMeansDenoisingMulti function for colored images sequencesParameters:srcImgs - Input 8-bit 3-channel images sequence. All images should have the same type and size.imgToDenoiseIndex - Target image to denoise index in srcImgs sequencetemporalWindowSize - Number of surrounding images to use for target image denoising. Should be odd. Images from imgToDenoiseIndex - temporalWindowSize / 2 to imgToDenoiseIndex - temporalWindowSize / 2 from srcImgs will be used to denoise srcImgs[imgToDenoiseIndex] image.dst - Output image with the same size and type as srcImgs images.templateWindowSize - Size in pixels of the template patch that is used to compute weights. Should be odd. Recommended value 7 pixelssearchWindowSize - Size in pixels of the window that is used to compute weighted average for given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Parameter regulating filter strength for luminance component. Bigger h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noise.hColor - The same as h but for color components. The function converts images to CIELAB colorspace and then separately denoise L and AB components with given h parameters using fastNlMeansDenoisingMulti function.

#### fastNlMeansDenoisingMulti public static void fastNlMeansDenoisingMulti(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize) Modification of fastNlMeansDenoising function for images sequence where consecutive images have been captured in small period of time. For example video. This version of the function is for grayscale images or for manual manipulation with colorspaces. For more details see <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.131.6394>Parameters:srcImgs - Input 8-bit 1-channel, 2-channel, 3-channel or 4-channel images sequence. All images should have the same type and size.imgToDenoiseIndex - Target image to denoise index in srcImgs sequencetemporalWindowSize - Number of surrounding images to use for target image denoising. Should be odd. Images from imgToDenoiseIndex - temporalWindowSize / 2 to imgToDenoiseIndex - temporalWindowSize / 2 from srcImgs will be used to denoise srcImgs[imgToDenoiseIndex] image.dst - Output image with the same size and type as srcImgs images. Should be odd. Recommended value 7 pixels given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixels perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noise

#### fastNlMeansDenoisingMulti public static void fastNlMeansDenoisingMulti(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, float h) Modification of fastNlMeansDenoising function for images sequence where consecutive images have been captured in small period of time. For example video. This version of the function is for grayscale images or for manual manipulation with colorspaces. For more details see <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.131.6394>Parameters:srcImgs - Input 8-bit 1-channel, 2-channel, 3-channel or 4-channel images sequence. All images should have the same type and size.imgToDenoiseIndex - Target image to denoise index in srcImgs sequencetemporalWindowSize - Number of surrounding images to use for target image denoising. Should be odd. Images from imgToDenoiseIndex - temporalWindowSize / 2 to imgToDenoiseIndex - temporalWindowSize / 2 from srcImgs will be used to denoise srcImgs[imgToDenoiseIndex] image.dst - Output image with the same size and type as srcImgs images. Should be odd. Recommended value 7 pixels given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Parameter regulating filter strength. Bigger h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noise

#### fastNlMeansDenoisingMulti public static void fastNlMeansDenoisingMulti(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, float h, int templateWindowSize) Modification of fastNlMeansDenoising function for images sequence where consecutive images have been captured in small period of time. For example video. This version of the function is for grayscale images or for manual manipulation with colorspaces. For more details see <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.131.6394>Parameters:srcImgs - Input 8-bit 1-channel, 2-channel, 3-channel or 4-channel images sequence. All images should have the same type and size.imgToDenoiseIndex - Target image to denoise index in srcImgs sequencetemporalWindowSize - Number of surrounding images to use for target image denoising. Should be odd. Images from imgToDenoiseIndex - temporalWindowSize / 2 to imgToDenoiseIndex - temporalWindowSize / 2 from srcImgs will be used to denoise srcImgs[imgToDenoiseIndex] image.dst - Output image with the same size and type as srcImgs images.templateWindowSize - Size in pixels of the template patch that is used to compute weights. Should be odd. Recommended value 7 pixels given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Parameter regulating filter strength. Bigger h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noise

#### fastNlMeansDenoisingMulti public static void fastNlMeansDenoisingMulti(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, float h, int templateWindowSize, int searchWindowSize) Modification of fastNlMeansDenoising function for images sequence where consecutive images have been captured in small period of time. For example video. This version of the function is for grayscale images or for manual manipulation with colorspaces. For more details see <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.131.6394>Parameters:srcImgs - Input 8-bit 1-channel, 2-channel, 3-channel or 4-channel images sequence. All images should have the same type and size.imgToDenoiseIndex - Target image to denoise index in srcImgs sequencetemporalWindowSize - Number of surrounding images to use for target image denoising. Should be odd. Images from imgToDenoiseIndex - temporalWindowSize / 2 to imgToDenoiseIndex - temporalWindowSize / 2 from srcImgs will be used to denoise srcImgs[imgToDenoiseIndex] image.dst - Output image with the same size and type as srcImgs images.templateWindowSize - Size in pixels of the template patch that is used to compute weights. Should be odd. Recommended value 7 pixelssearchWindowSize - Size in pixels of the window that is used to compute weighted average for given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Parameter regulating filter strength. Bigger h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noise

#### fastNlMeansDenoisingMulti public static void fastNlMeansDenoisingMulti(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) h) Modification of fastNlMeansDenoising function for images sequence where consecutive images have been captured in small period of time. For example video. This version of the function is for grayscale images or for manual manipulation with colorspaces. For more details see <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.131.6394>Parameters:srcImgs - Input 8-bit or 16-bit (only with NORM\_L1) 1-channel, 2-channel, 3-channel or 4-channel images sequence. All images should have the same type and size.imgToDenoiseIndex - Target image to denoise index in srcImgs sequencetemporalWindowSize - Number of surrounding images to use for target image denoising. Should be odd. Images from imgToDenoiseIndex - temporalWindowSize / 2 to imgToDenoiseIndex - temporalWindowSize / 2 from srcImgs will be used to denoise srcImgs[imgToDenoiseIndex] image.dst - Output image with the same size and type as srcImgs images. Should be odd. Recommended value 7 pixels given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Array of parameters regulating filter strength, either one parameter applied to all channels or one per channel in dst. Big h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noise

#### fastNlMeansDenoisingMulti public static void fastNlMeansDenoisingMulti(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) h, int templateWindowSize) Modification of fastNlMeansDenoising function for images sequence where consecutive images have been captured in small period of time. For example video. This version of the function is for grayscale images or for manual manipulation with colorspaces. For more details see <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.131.6394>Parameters:srcImgs - Input 8-bit or 16-bit (only with NORM\_L1) 1-channel, 2-channel, 3-channel or 4-channel images sequence. All images should have the same type and size.imgToDenoiseIndex - Target image to denoise index in srcImgs sequencetemporalWindowSize - Number of surrounding images to use for target image denoising. Should be odd. Images from imgToDenoiseIndex - temporalWindowSize / 2 to imgToDenoiseIndex - temporalWindowSize / 2 from srcImgs will be used to denoise srcImgs[imgToDenoiseIndex] image.dst - Output image with the same size and type as srcImgs images.templateWindowSize - Size in pixels of the template patch that is used to compute weights. Should be odd. Recommended value 7 pixels given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Array of parameters regulating filter strength, either one parameter applied to all channels or one per channel in dst. Big h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noise

#### fastNlMeansDenoisingMulti public static void fastNlMeansDenoisingMulti(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) h, int templateWindowSize, int searchWindowSize) Modification of fastNlMeansDenoising function for images sequence where consecutive images have been captured in small period of time. For example video. This version of the function is for grayscale images or for manual manipulation with colorspaces. For more details see <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.131.6394>Parameters:srcImgs - Input 8-bit or 16-bit (only with NORM\_L1) 1-channel, 2-channel, 3-channel or 4-channel images sequence. All images should have the same type and size.imgToDenoiseIndex - Target image to denoise index in srcImgs sequencetemporalWindowSize - Number of surrounding images to use for target image denoising. Should be odd. Images from imgToDenoiseIndex - temporalWindowSize / 2 to imgToDenoiseIndex - temporalWindowSize / 2 from srcImgs will be used to denoise srcImgs[imgToDenoiseIndex] image.dst - Output image with the same size and type as srcImgs images.templateWindowSize - Size in pixels of the template patch that is used to compute weights. Should be odd. Recommended value 7 pixelssearchWindowSize - Size in pixels of the window that is used to compute weighted average for given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Array of parameters regulating filter strength, either one parameter applied to all channels or one per channel in dst. Big h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noise

#### fastNlMeansDenoisingMulti public static void fastNlMeansDenoisingMulti(java.util.List<[Mat](http://docs.google.com/org/opencv/core/Mat.html)> srcImgs, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, int imgToDenoiseIndex, int temporalWindowSize, [MatOfFloat](http://docs.google.com/org/opencv/core/MatOfFloat.html) h, int templateWindowSize, int searchWindowSize, int normType) Modification of fastNlMeansDenoising function for images sequence where consecutive images have been captured in small period of time. For example video. This version of the function is for grayscale images or for manual manipulation with colorspaces. For more details see <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.131.6394>Parameters:srcImgs - Input 8-bit or 16-bit (only with NORM\_L1) 1-channel, 2-channel, 3-channel or 4-channel images sequence. All images should have the same type and size.imgToDenoiseIndex - Target image to denoise index in srcImgs sequencetemporalWindowSize - Number of surrounding images to use for target image denoising. Should be odd. Images from imgToDenoiseIndex - temporalWindowSize / 2 to imgToDenoiseIndex - temporalWindowSize / 2 from srcImgs will be used to denoise srcImgs[imgToDenoiseIndex] image.dst - Output image with the same size and type as srcImgs images.templateWindowSize - Size in pixels of the template patch that is used to compute weights. Should be odd. Recommended value 7 pixelssearchWindowSize - Size in pixels of the window that is used to compute weighted average for given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixelsh - Array of parameters regulating filter strength, either one parameter applied to all channels or one per channel in dst. Big h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noisenormType - Type of norm used for weight calculation. Can be either NORM\_L2 or NORM\_L1

#### illuminationChange public static void illuminationChange([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Applying an appropriate non-linear transformation to the gradient field inside the selection and then integrating back with a Poisson solver, modifies locally the apparent illumination of an image.Parameters:src - Input 8-bit 3-channel image.mask - Input 8-bit 1 or 3-channel image.dst - Output image with the same size and type as src. This is useful to highlight under-exposed foreground objects or to reduce specular reflections.

#### illuminationChange public static void illuminationChange([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float alpha) Applying an appropriate non-linear transformation to the gradient field inside the selection and then integrating back with a Poisson solver, modifies locally the apparent illumination of an image.Parameters:src - Input 8-bit 3-channel image.mask - Input 8-bit 1 or 3-channel image.dst - Output image with the same size and type as src.alpha - Value ranges between 0-2. This is useful to highlight under-exposed foreground objects or to reduce specular reflections.

#### illuminationChange public static void illuminationChange([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float alpha, float beta) Applying an appropriate non-linear transformation to the gradient field inside the selection and then integrating back with a Poisson solver, modifies locally the apparent illumination of an image.Parameters:src - Input 8-bit 3-channel image.mask - Input 8-bit 1 or 3-channel image.dst - Output image with the same size and type as src.alpha - Value ranges between 0-2.beta - Value ranges between 0-2. This is useful to highlight under-exposed foreground objects or to reduce specular reflections.

#### inpaint public static void inpaint([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) inpaintMask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, double inpaintRadius, int flags) Restores the selected region in an image using the region neighborhood.Parameters:src - Input 8-bit, 16-bit unsigned or 32-bit float 1-channel or 8-bit 3-channel image.inpaintMask - Inpainting mask, 8-bit 1-channel image. Non-zero pixels indicate the area that needs to be inpainted.dst - Output image with the same size and type as src .inpaintRadius - Radius of a circular neighborhood of each point inpainted that is considered by the algorithm.flags - Inpainting method that could be cv::INPAINT\_NS or cv::INPAINT\_TELEA The function reconstructs the selected image area from the pixel near the area boundary. The function may be used to remove dust and scratches from a scanned photo, or to remove undesirable objects from still images or video. See <http://en.wikipedia.org/wiki/Inpainting> for more details. **Note:**

* + - * An example using the inpainting technique can be found at opencv\_source\_code/samples/cpp/inpaint.cpp
      * (Python) An example using the inpainting technique can be found at opencv\_source\_code/samples/python/inpaint.py

#### pencilSketch public static void pencilSketch([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst2) Pencil-like non-photorealistic line drawingParameters:src - Input 8-bit 3-channel image.dst1 - Output 8-bit 1-channel image.dst2 - Output image with the same size and type as src.

#### pencilSketch public static void pencilSketch([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst2, float sigma\_s) Pencil-like non-photorealistic line drawingParameters:src - Input 8-bit 3-channel image.dst1 - Output 8-bit 1-channel image.dst2 - Output image with the same size and type as src.sigma\_s - %Range between 0 to 200.

#### pencilSketch public static void pencilSketch([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst2, float sigma\_s, float sigma\_r) Pencil-like non-photorealistic line drawingParameters:src - Input 8-bit 3-channel image.dst1 - Output 8-bit 1-channel image.dst2 - Output image with the same size and type as src.sigma\_s - %Range between 0 to 200.sigma\_r - %Range between 0 to 1.

#### pencilSketch public static void pencilSketch([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst2, float sigma\_s, float sigma\_r, float shade\_factor) Pencil-like non-photorealistic line drawingParameters:src - Input 8-bit 3-channel image.dst1 - Output 8-bit 1-channel image.dst2 - Output image with the same size and type as src.sigma\_s - %Range between 0 to 200.sigma\_r - %Range between 0 to 1.shade\_factor - %Range between 0 to 0.1.

#### seamlessClone public static void seamlessClone([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, [Point](http://docs.google.com/org/opencv/core/Point.html) p, [Mat](http://docs.google.com/org/opencv/core/Mat.html) blend, int flags) Image editing tasks concern either global changes (color/intensity corrections, filters, deformations) or local changes concerned to a selection. Here we are interested in achieving local changes, ones that are restricted to a region manually selected (ROI), in a seamless and effortless manner. The extent of the changes ranges from slight distortions to complete replacement by novel content CITE: PM03 .Parameters:src - Input 8-bit 3-channel image.dst - Input 8-bit 3-channel image.mask - Input 8-bit 1 or 3-channel image.p - Point in dst image where object is placed.blend - Output image with the same size and type as dst.flags - Cloning method that could be cv::NORMAL\_CLONE, cv::MIXED\_CLONE or cv::MONOCHROME\_TRANSFER

#### stylization public static void stylization([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) Stylization aims to produce digital imagery with a wide variety of effects not focused on photorealism. Edge-aware filters are ideal for stylization, as they can abstract regions of low contrast while preserving, or enhancing, high-contrast features.Parameters:src - Input 8-bit 3-channel image.dst - Output image with the same size and type as src.

#### stylization public static void stylization([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float sigma\_s) Stylization aims to produce digital imagery with a wide variety of effects not focused on photorealism. Edge-aware filters are ideal for stylization, as they can abstract regions of low contrast while preserving, or enhancing, high-contrast features.Parameters:src - Input 8-bit 3-channel image.dst - Output image with the same size and type as src.sigma\_s - %Range between 0 to 200.

#### stylization public static void stylization([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float sigma\_s, float sigma\_r) Stylization aims to produce digital imagery with a wide variety of effects not focused on photorealism. Edge-aware filters are ideal for stylization, as they can abstract regions of low contrast while preserving, or enhancing, high-contrast features.Parameters:src - Input 8-bit 3-channel image.dst - Output image with the same size and type as src.sigma\_s - %Range between 0 to 200.sigma\_r - %Range between 0 to 1.

#### textureFlattening public static void textureFlattening([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst) By retaining only the gradients at edge locations, before integrating with the Poisson solver, one washes out the texture of the selected region, giving its contents a flat aspect. Here Canny Edge %Detector is used.Parameters:src - Input 8-bit 3-channel image.mask - Input 8-bit 1 or 3-channel image.dst - Output image with the same size and type as src. **Note:** The algorithm assumes that the color of the source image is close to that of the destination. This assumption means that when the colors don't match, the source image color gets tinted toward the color of the destination image.

#### textureFlattening public static void textureFlattening([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float low\_threshold) By retaining only the gradients at edge locations, before integrating with the Poisson solver, one washes out the texture of the selected region, giving its contents a flat aspect. Here Canny Edge %Detector is used.Parameters:src - Input 8-bit 3-channel image.mask - Input 8-bit 1 or 3-channel image.dst - Output image with the same size and type as src.low\_threshold - %Range from 0 to 100. **Note:** The algorithm assumes that the color of the source image is close to that of the destination. This assumption means that when the colors don't match, the source image color gets tinted toward the color of the destination image.

#### textureFlattening public static void textureFlattening([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float low\_threshold, float high\_threshold) By retaining only the gradients at edge locations, before integrating with the Poisson solver, one washes out the texture of the selected region, giving its contents a flat aspect. Here Canny Edge %Detector is used.Parameters:src - Input 8-bit 3-channel image.mask - Input 8-bit 1 or 3-channel image.dst - Output image with the same size and type as src.low\_threshold - %Range from 0 to 100.high\_threshold - Value > 100. **Note:** The algorithm assumes that the color of the source image is close to that of the destination. This assumption means that when the colors don't match, the source image color gets tinted toward the color of the destination image.

#### textureFlattening public static void textureFlattening([Mat](http://docs.google.com/org/opencv/core/Mat.html) src, [Mat](http://docs.google.com/org/opencv/core/Mat.html) mask, [Mat](http://docs.google.com/org/opencv/core/Mat.html) dst, float low\_threshold, float high\_threshold, int kernel\_size) By retaining only the gradients at edge locations, before integrating with the Poisson solver, one washes out the texture of the selected region, giving its contents a flat aspect. Here Canny Edge %Detector is used.Parameters:src - Input 8-bit 3-channel image.mask - Input 8-bit 1 or 3-channel image.dst - Output image with the same size and type as src.low\_threshold - %Range from 0 to 100.high\_threshold - Value > 100.kernel\_size - The size of the Sobel kernel to be used. **Note:** The algorithm assumes that the color of the source image is close to that of the destination. This assumption means that when the colors don't match, the source image color gets tinted toward the color of the destination image.

* [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/photo/MergeRobertson.html)
* [Next Class](http://docs.google.com/org/opencv/photo/Tonemap.html)
* [Frames](http://docs.google.com/index.html?org/opencv/photo/Photo.html)
* [No Frames](http://docs.google.com/Photo.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* [Field](#3znysh7) |
* [Constr](#2et92p0) |
* [Method](#tyjcwt)
* Detail:
* [Field](#1t3h5sf) |
* [Constr](#44sinio) |
* [Method](#z337ya)

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