OpenCV2 Functions

OpenCV Documentation

http://docs.opencv.org/index.html

Common Functions

- cv::imread()
- cv::imshow()
- cv::imwrite()
- cv::namedWindow()
- cv::waitKey()

Inverse Function (1/2)

Inverse Function (2/2)

bitwise_not

Inverts every bit of an array.

C++: void **bitwise_not**(InputArray **src**, OutputArray **dst**, InputArray **mask**=noArray())

Python: cv2.**bitwise_not**(src[, dst[, mask]]) → dst

C: void **cvNot**(const CvArr* **src**, CvArr* **dst**)

Python: cv.**Not**(src, dst) → None

Parameters:
- **src** – input array.
- **dst** – output array that has the same size and type as the input array.
- **mask** – optional operation mask, 8-bit single channel array, that specifies elements of the output array to be changed.

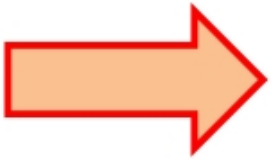
The function calculates per-element bit-wise inversion of the input array:

$$\texttt{dst}(I) = \neg \texttt{src}(I)$$

In case of a floating-point input array, its machine-specific bit representation (usually IEEE754-compliant) is used for the operation. In case of multi-channel arrays, each channel is processed independently.

```
Mat Gray_Lena=imread("lena_8bit.bmp",0);
Mat Inv_Lena;
bitwise_not(Gray_Lena,Inv_Lena);
```

RGB ↔ Gray



cvtColor Function

cvtColor

Converts an image from one color space to another.

C++: void **cvtColor**(InputArray **src**, OutputArray **dst**, int **code**, int **dstCn**=0)

Parameters:
- **src** – input image: 8-bit unsigned, 16-bit unsigned (`CV_16UC...`), or single-precision floating-point.
- **dst** – output image of the same size and depth as `src`.
- **code** – color space conversion code (see the description below).
- **dstCn** – number of channels in the destination image; if the parameter is 0, the number of the channels is derived automatically from `src` and `code` .

RGB ↔ GRAY

– CV_BGR2GRAY, CV_RGB2GRAY, CV_GRAY2BGR, CV_GRAY2RGB

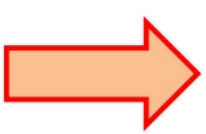
RGB ↔ YCrCb

– CV_BGR2YCrCb, CV_RGB2YCrCb, CV_YCrCb2BGR, CV_YCrCb2RGB

…

```
cvtColor(RGB_Lena,Gray_Lena,CV_RGB2GRAY);
```

Color Image → RGB Planes



Split & Merge Function

split ¶

Divides a multi-channel array into several single-channel arrays.

C++: void **split**(const Mat& **src**, Mat* **mvbegin**)

C++: void **split**(InputArray **m**, OutputArrayOfArrays **mv**)

merge ¶

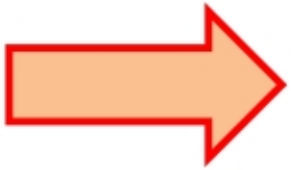
Creates one multichannel array out of several single-channel ones.

C++: void **merge**(const Mat* **mv**, size_t **count**, OutputArray **dst**)

C++: void **merge**(InputArrayOfArrays **mv**, OutputArray **dst**)

```
Mat RGB_Lena=imread("lena_std2_24b.bmp",1);

vector<Mat> Channel;
split(RGB_Lena,Channel);

Mat Zero=Mat::zeros(RGB_Lena.rows,RGB_Lena.cols,CV_8UC1);

vector<Mat> Blue_Channel,Green_Channel,Red_Channel;
Blue_Channel.push_back(Channel[0]);
Blue_Channel.push_back(Zero);
Blue_Channel.push_back(Zero);

Green_Channel.push_back(Zero);
Green_Channel.push_back(Channel[1]);
Green_Channel.push_back(Zero);

Red_Channel.push_back(Zero);
Red_Channel.push_back(Zero);
Red_Channel.push_back(Channel[2]);

Mat Blue_Result,Green_Result,Red_Result;
merge(Blue_Channel,Blue_Result);
merge(Green_Channel,Green_Result);
merge(Red_Channel,Red_Result);
```

Image → Binary Image



threshold Function

threshold

Applies a fixed-level threshold to each array element.

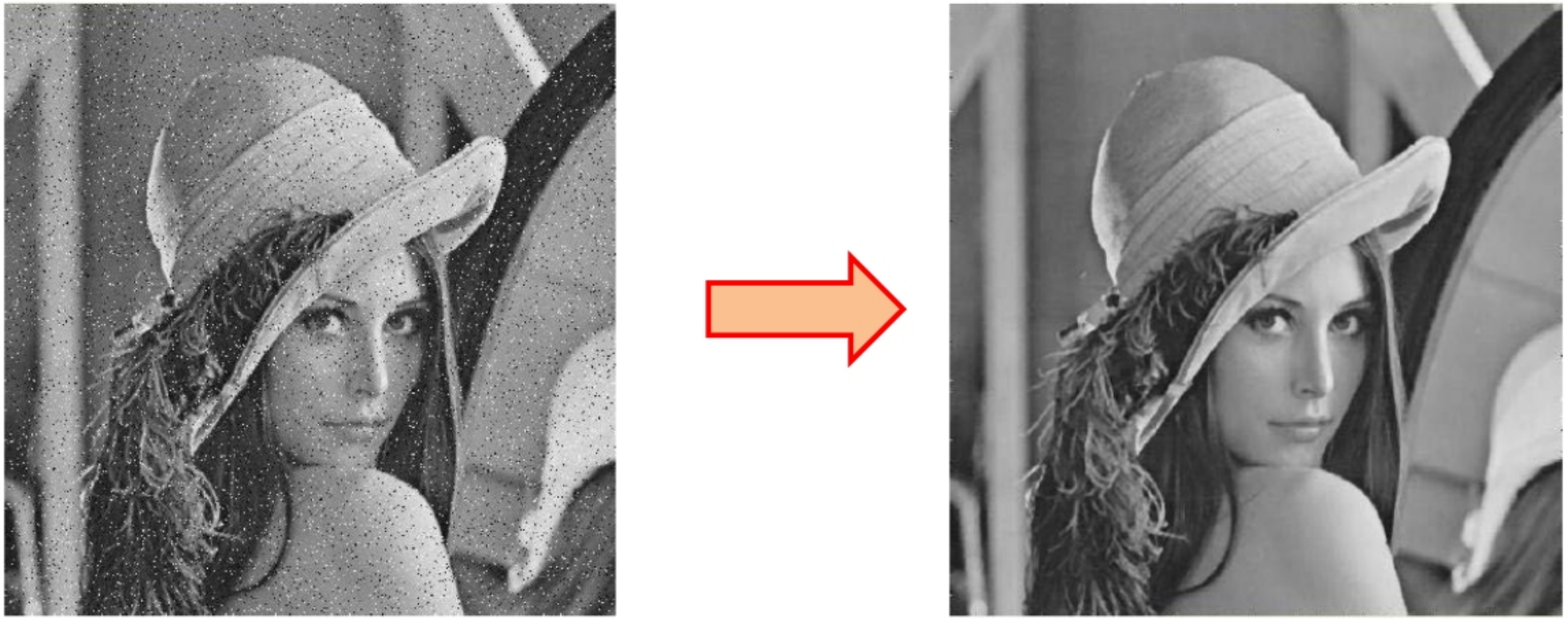
C++: double **threshold**(InputArray **src**, OutputArray **dst**, double **thresh**, double **maxval**, int **type**)

Parameters:
- **src** – input array (single-channel, 8-bit or 32-bit floating point).
- **dst** – output array of the same size and type as `src`.
- **thresh** – threshold value.
- **maxval** – maximum value to use with the `THRESH_BINARY` and `THRESH_BINARY_INV` thresholding types.
- **type** – thresholding type (see the details below).

THRESH_BINARY $\texttt{dst}(x,y) = \begin{cases} \texttt{maxval} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$

```
Mat Gray_Lena=imread("lena_8bit.bmp",0);

Mat Binary_Result;
threshold(Gray_Lena,Binary_Result,125,255,THRESH_BINARY);
```

Median Filter



medianBlur Function

medianBlur

Blurs an image using the median filter.

C++: void **medianBlur**(InputArray **src**, OutputArray **dst**, int **ksize**)

Parameters:

- **src** – input 1-, 3-, or 4-channel image; when `ksize` is 3 or 5, the image depth should be `CV_8U`, `CV_16U`, or `CV_32F`, for larger aperture sizes, it can only be `CV_8U`.
- **dst** – destination array of the same size and type as `src`.
- **ksize** – aperture linear size; it must be odd and greater than 1, for example: 3, 5, 7 ...

```
Mat Gray_Lena=imread("lena_8bit.bmp",0);

Mat Result;
medianBlur(Gray_Lena,Result,3);
```

Homework!!

- 1. filter2D - > Low pass filter
- 2. Sobel (convertScaleAbs)
- 3. Dilation
- 4. Erosion