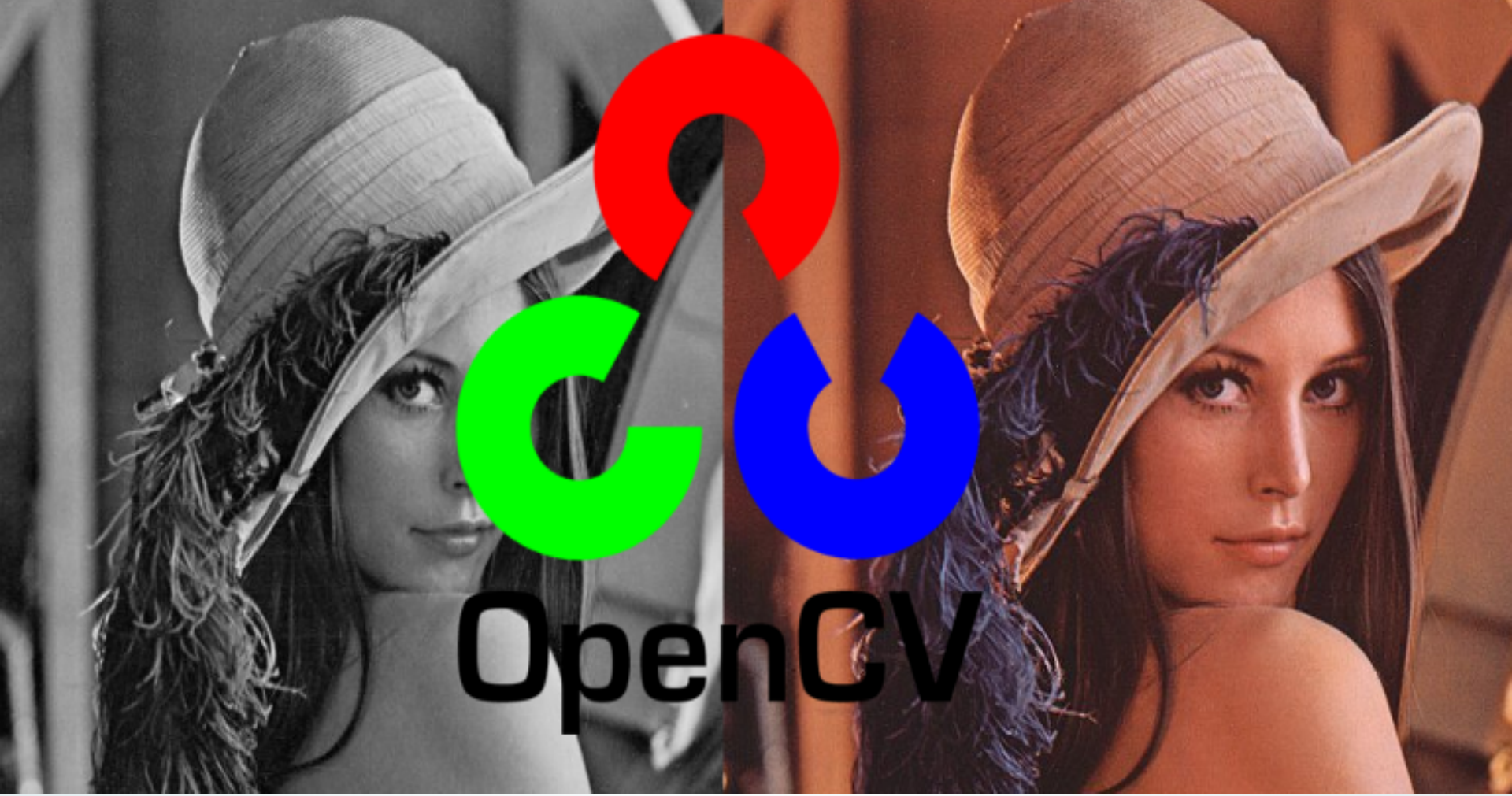


OpenCV: Modules and Functions

Ghada Zamzmi

Slides Credits Marvin Smith and Anne Solberg (U of UiO)

Outline

- Introduction
- Installation Guides
- OpenCV Modules
- Image processing Functions
    - Thresholding
    - Edge Detection
    - Hough Transform
    - Smoothing
    - Color Space

Introduction

- OpenCV is a library for **image processing** and **computer vision**
- It has several hundreds of built-in functions
- **Open Source and Free**
- Primary interface of OpenCV is in C++
- There are also full interfaces of C, Python and JAVA
- It runs on **Windows**, **Linux, and Mac**

Installation Guides for Different Platforms

- Windows:
  - http://docs.opencv.org/2.4/doc/tutorials/introduction/windows_install/windows_install.html
  - http://opencv-srf.blogspot.com/2013/05/installing-configuring-opencv-with-vs.html
- Linux:
  - http://docs.opencv.org/2.4/doc/tutorials/introduction/linux_install/linux_install.html
  - http://docs.opencv.org/2.4/doc/tutorials/introduction/linux_eclipse/linux_eclipse.html
- MAC:
  - http://blogs.wcode.org/2014/10/howto-install-build-and-use-opencv-macosx-10-10/
  - http://mac-opencv-projects.blogspot.com/2014/01/installing-opencv-on-mac-os-x-1091.html

If you have questions about installation, email me!

OpenCV Modules

- OpenCV is a modular structure library
- It has several modules such as:
  - **core:** basic OpenCV data structures
  - **highgui:** image I/O operations
  - **imgproc**: image processing functions (e.g., filtering and thresholding)
  - ….. Other modules

OpenCV Modules

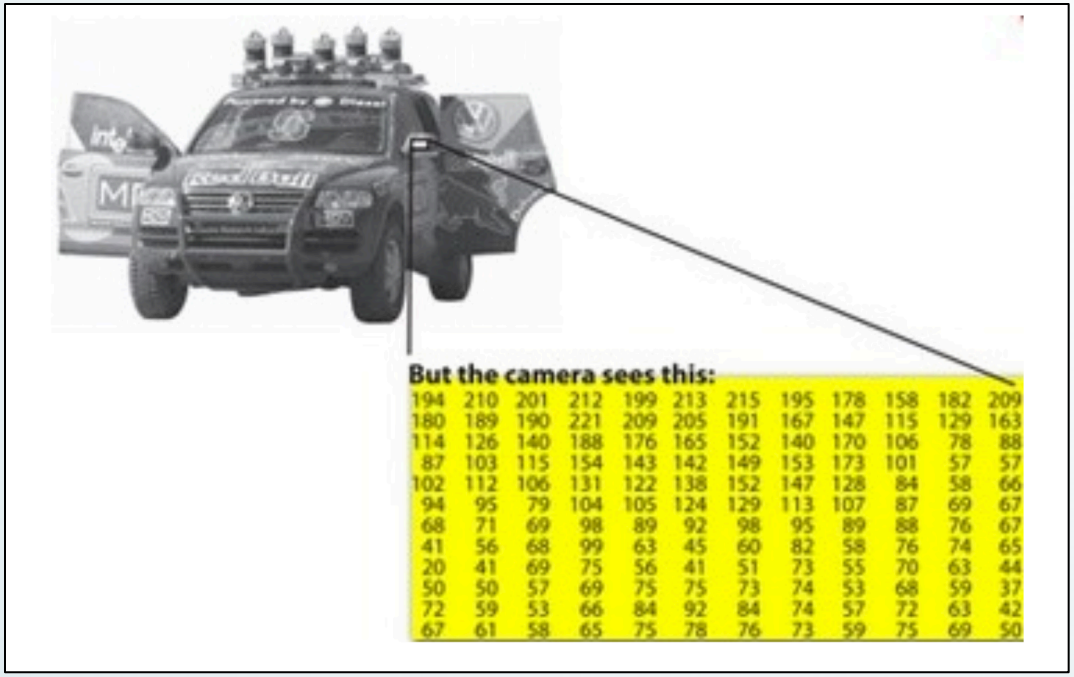
OpenCV Modules: core Module

- Core module defines the basic OpenCV data structures

- Basic data structures include:
  - **Point:** 2D point (x,y)
  - **Rect:** 2D rectangle object
  - **Vec**: row or column vector
  - **Mat:** Matrix object

Mat: The basic Image Container

- Mat is the primary data structure in OpenCV
- It is used to store an image as numerical matrix
- Each cell of the matrix represents the intensity of a specific pixel.



Remember:

rows is your y coordinate
and
cols is your x coordinate

Mat Data Structure (cont.)

- **Functions:**
  - Mat.**at**<imagetype>(x, y)[channel] – returns the intensity of a pixel at x and y coordinates.
  - Mat.**channels**() - returns the image's number of channels.
  - Mat.**clone**() - returns a copy of the image
  - Mat.**size**() - returns the SIZE of an image.
  - Mat.**type**() - returns the TYPE of an image.

highgui Module: Image I/O Operations

- OpenCV provides simple and useful ways to read and write images.

- Examples

```
//Read an image
Mat image = imread( <filename>)

//Write an image
imwrite( <string filename> , image );

//Output image to window
imshow( <window name> , <image Mat to show> );

//pause program for input
key = waitKey( 0 );
```

Example Code

```
//header of modules
#include <cv.h>
#include <highgui.h>

int main(int argc, char* argv[ ]){
   Mat image = imread(argv[1]);

   namedWindow("Lena");
   imshow("Lena",image);
   waitKey(0);
   return 0;
}
```

This program will load and show an image

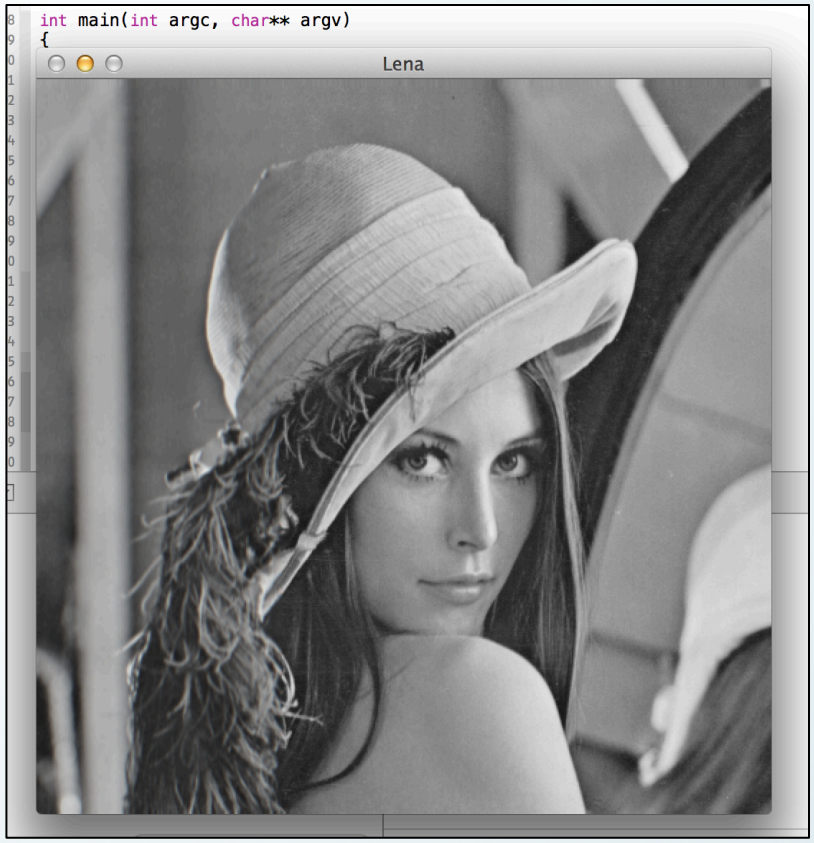


Image Format in OpenCV

- If the image cannot be read because of:
  - missing file
  - improper permissions
  - unsupported or invalid format
- imread function returns an empty matrix (Mat::data==NULL)

Image Format in OpenCV

- Currently, the following file formats are supported:
  - Windows bitmaps - *.bmp, *.dib
  - JPEG files - *.jpeg, *.jpg, *.jpe
  - JPEG 2000 files - *.jp2
  - Portable Network Graphics - *.png
  - WebP - *.webp
  - Portable image format - *.pbm, *.pgm, *.ppm
  - Sun rasters - *.sr, *.ras
  - TIFF files - *.tiff, *.tif

imageproc Module

- Image processing module has many functions such as:
  - Thresholding
  - Image smoothing
  - Edge detections
  - Color space conversion
  - Hough Transform
  - Geometrical image transformations
  - Histogram modifications
  - an so on…

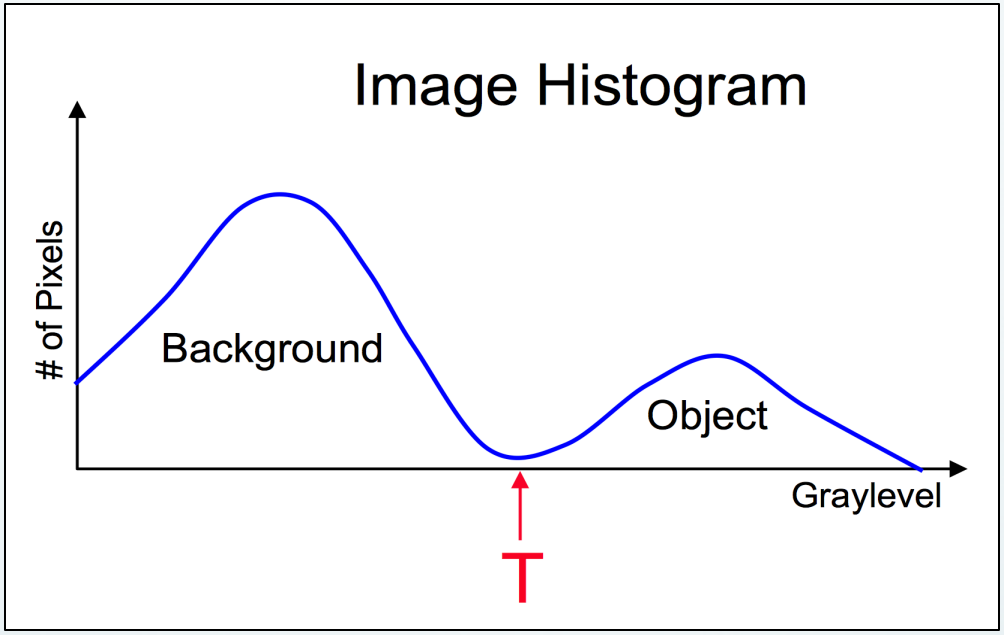
Thresholding

- **Background:**
  - It's a point operator.
  - The simplest method of segmentation.
  - Reject those pixels above or below some value (i.e., threshold) while keeping the others.

Thresholding (cont.)

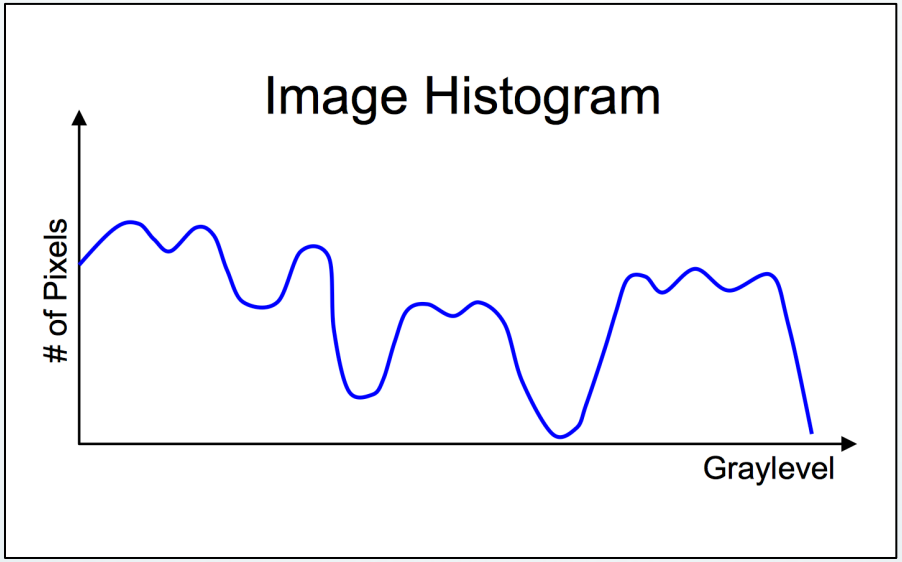
Global Thresholding =

Choose threshold **T** that separates object from background.



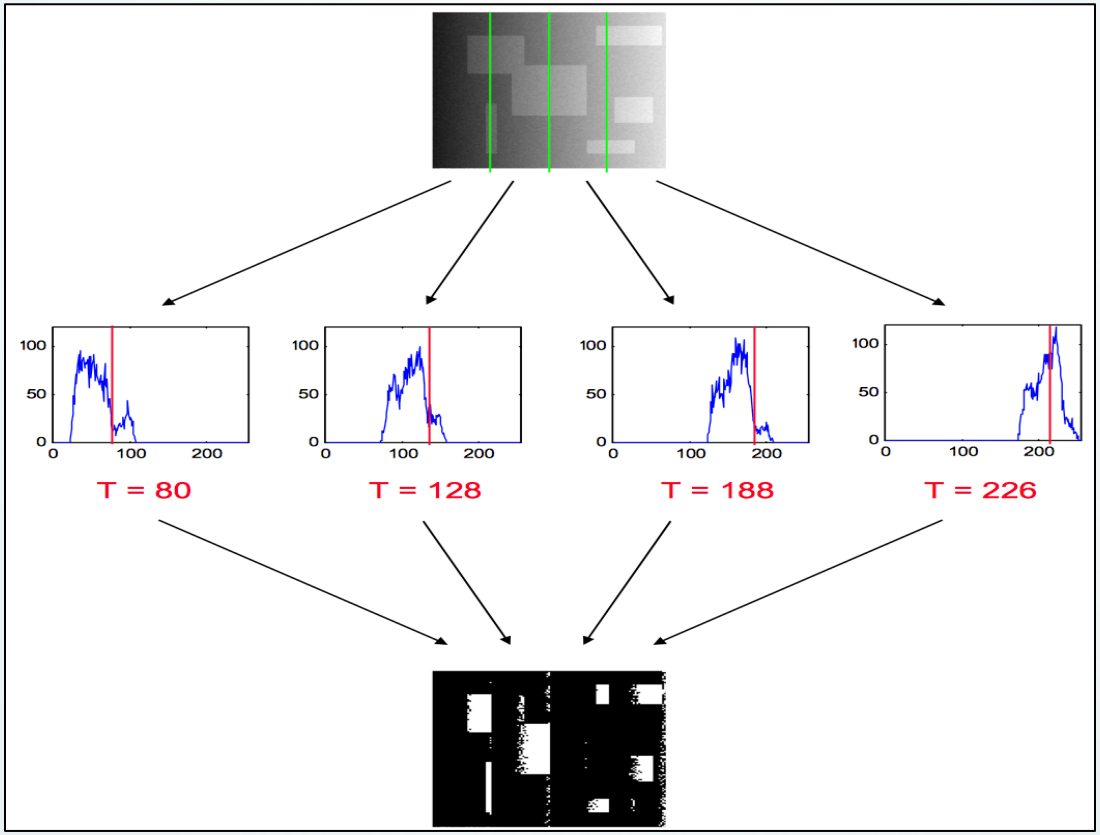
Thresholding (cont.)

- The Simple global thresholding is not always possible.
- What if the image histogram due to noise or large intensities variations looks like:



Thresholding (cont.)

- **Local thresholding:** divide the image into regions and perform thresholding in each region.



Thresholding (cont.)

- **Other Methods:**
  - Adaptive Thresholding.
  - Otsu's Thresholding.
  - Region growing
  - Graph cut and so on…

OpenCV imageproc Module:
Threshold Function

- void threshold(src, dst, thresh, maxVal, thresold_type);

  Parameters:
  - **src** – source image
  - **dst** – Destination image
  - **thresh** - a user specified threshold
  - **maxValue** – The new pixel intensity
  - **thresholdType** – Thresholding type. It must be from the supported types

- **Example:**

threshold(src, dst, 100, 255, THRESH_BINARY);

OpenCV imageproc Module: Threshold Function

- Below is a list of threshold_types in OpenCV:

- **THRESH_BINARY**

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

- **THRESH_BINARY_INV**

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

- **THRESH_TRUNC**

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

- **THRESH_TOZERO**

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

- **THRESH_TOZERO_INV**

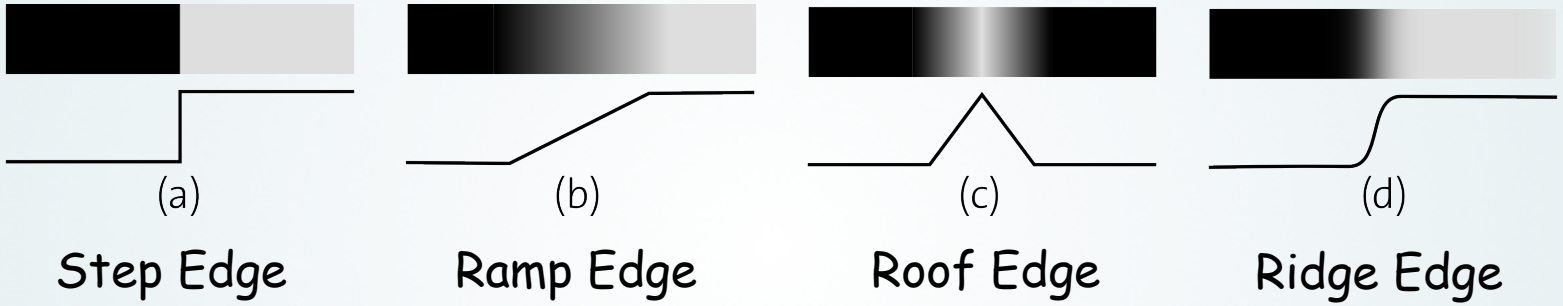
$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

Edge Detections

- Another way for image segmentation.
- Convert a 2D image into a set of curves (i.e., edges) and background.
- Edges are significant local changes of intensity in an image.
- Edge Types include: step edge, ramp edge, ridge edge, and roof edge.

Edge Detections

- Examples of edge types:



- Gradient Operators to detect edges.

change in the pixel value ⟶ large gradient

Edge Detections

- Examples of gradient operators:

$$\text{Sobel}: \quad H_1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad H_2 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$\text{Prewitt}: \quad H_1 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad H_2 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\text{Roberts}: \quad H_1 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad H_2 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

OpenCV imageproc Module:
Edge Detections

- void **Sobel**(src, dst, x_order, y_order, ksize, bordertype)

  Parameters:

  – **src** – input image.

  – **dst** – output image.

  – **X_order and y_order** – the direction of the gradient; x_order =1 and y_order = 0 calculate the gradient in x direction

  – **ksize** – size of the Sobel kernel (H); it must be 1, 3, 5, or 7.

  – **borderType** – pixel extrapolation method

OpenCV imageproc Module:
Edge Detections

- Example: Sobel operator (*th=48*)

Original Image

Horizontal Edge

Vertical Edge

OpenCV has functions for other gradient operators.

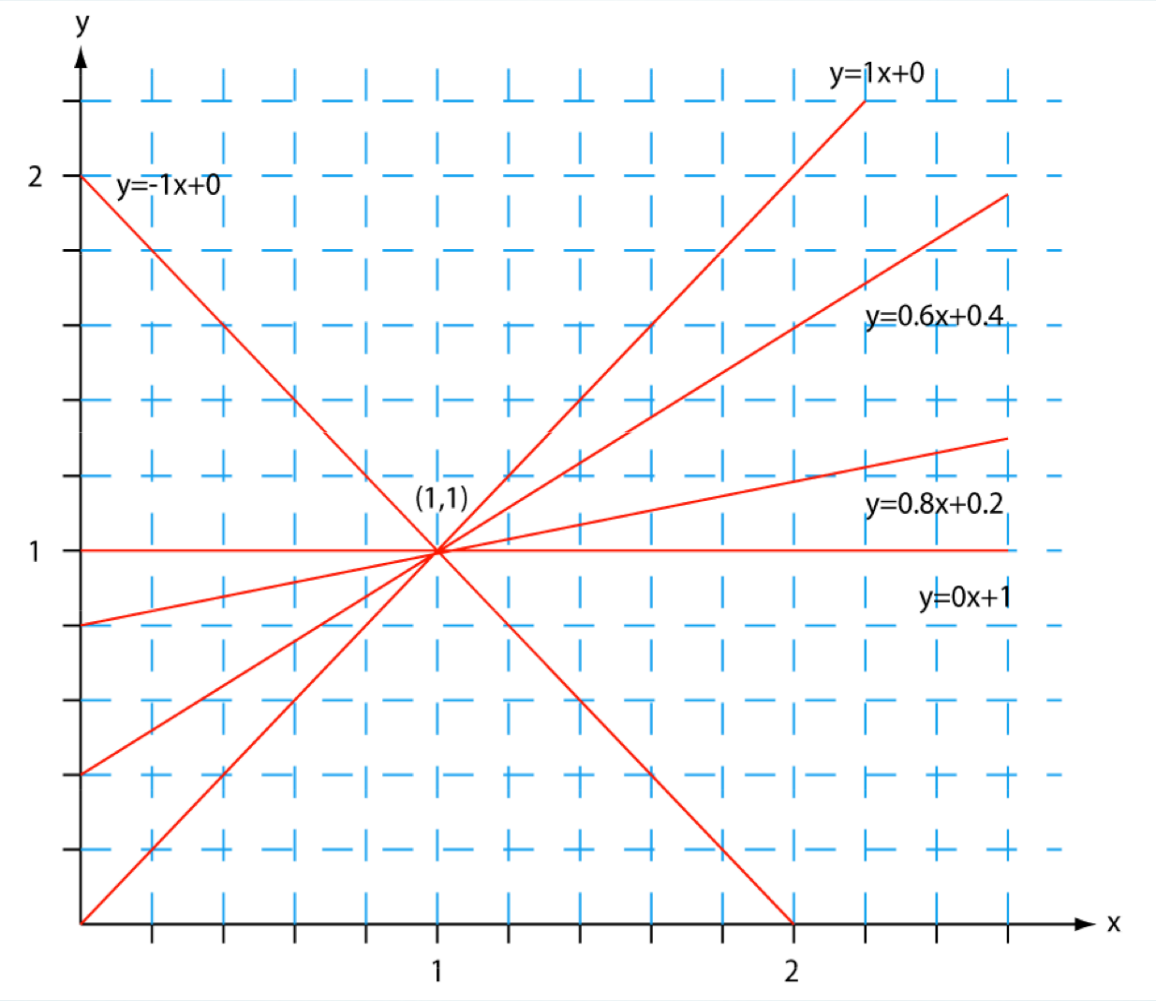
Hough Transform

- It is a transform for direct object recognition (e.g., lines, circles)
- It can detect any parametric curves if their parametric equation is known
- **Applied on detected edges image**

Hough Line Transform

- **Input:** the thresholded edge detection image
- We want to find the set of pixels in the input image that make up straight lines
- Regard a point $(x_i; y_i)$ and a straight line $y_i = ax_i + b$,
  - There can be many lines passing through the point (x_i,y_i)
  - They all satisfy the equation for some set of parameters (a,b)

Hough Line Transform



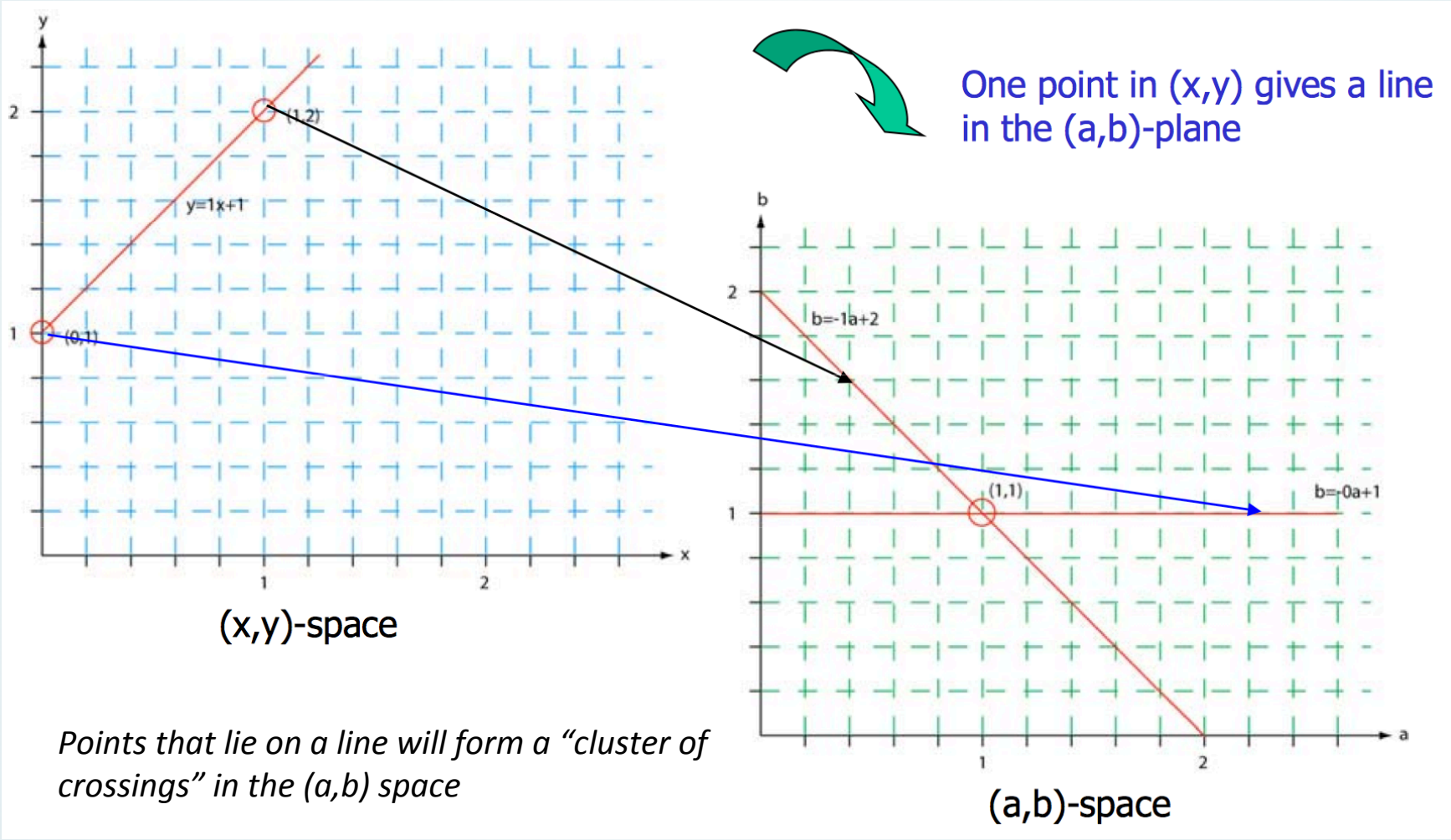
Hough Line Transform

- The equation ($y_i = ax_i + b$) can be rewritten as:

$$b = -xa + y$$

- We now consider x and y as parameters and a and b as variables
- This is a line in (a,b) space parameterized by x and y
  - So: a single point in xy-space gives a line in (a,b) space
- Another point (x,y) will give rise to another line in (a,b) space

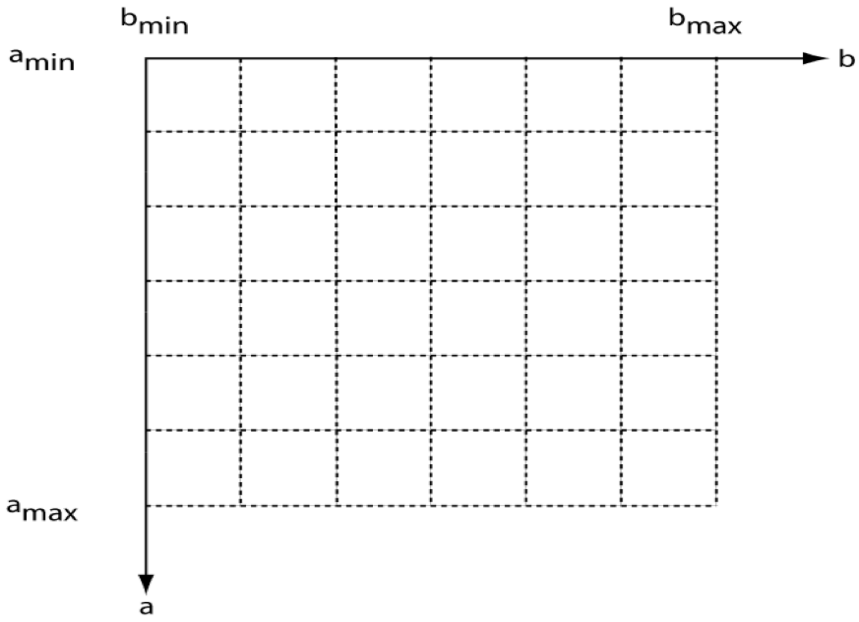
Hough Line Transform



Points that lie on a line will form a "cluster of crossings" in the (a,b) space

Hough Transform – Algorithm

- Quantize the parameter space (a,b), that is, divide it into cells.



Hough accumulator cells

Hough Transform – Algorithm

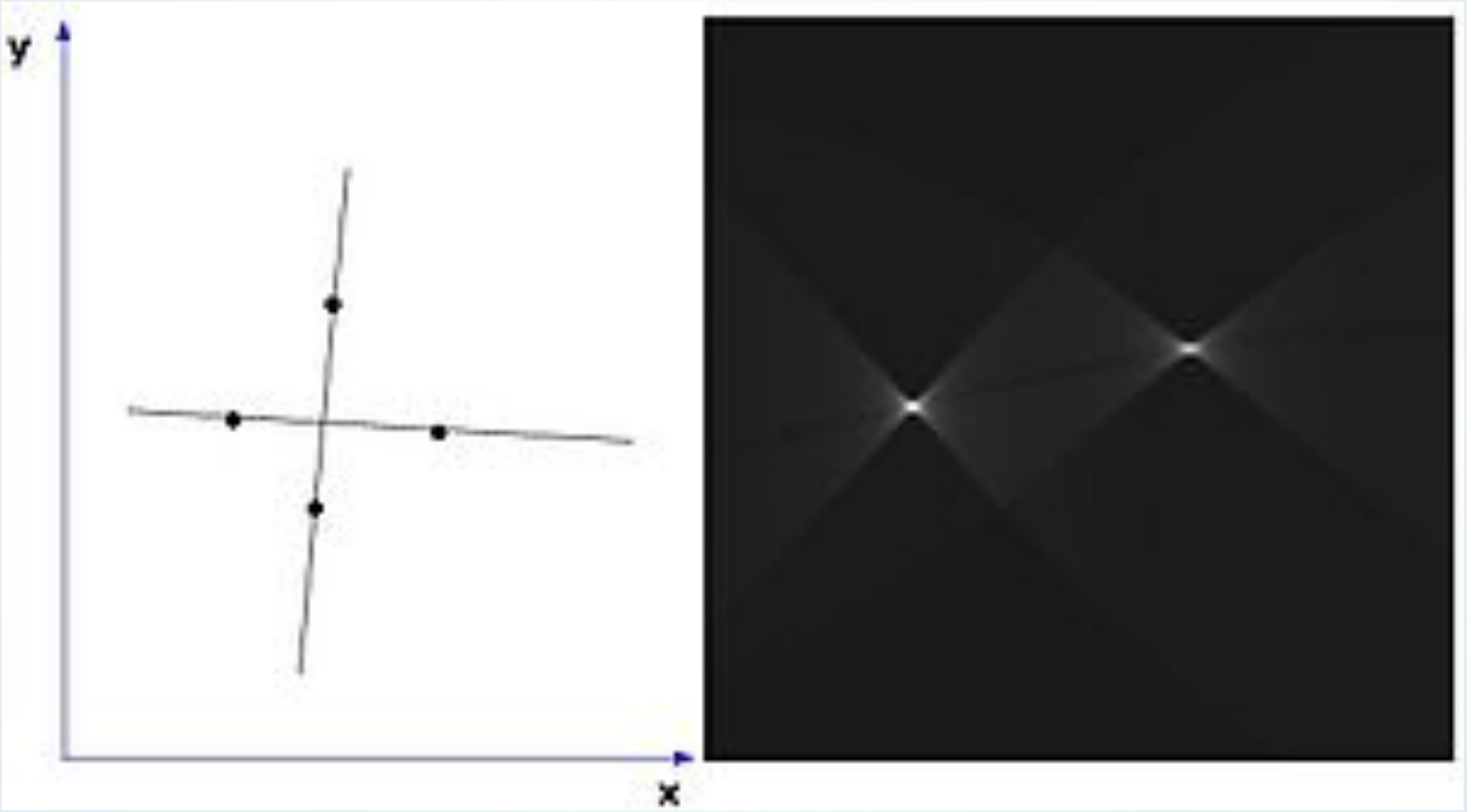
- Count the number of times a line intersects a given cell
  - For each point (x,y) with value 1 in the binary image, find the value of (a,b) in the parameter space
  - Increase the value of the accumulator for this (a,b) point.
  - Proceed with the next point in the image.

- Set a threshold and compare the number of votes for each cell to the threshold
  - Cells that exceed the threshold correspond to lines in (x,y) space

Hough Transform - Algorithm

XY Space

ab or Hough Space



OpenCV imageproc Module: Hough Transform – Lines

- void **HoughLines**(src, put_lines, rho, theta, thresh)

  Parameters:

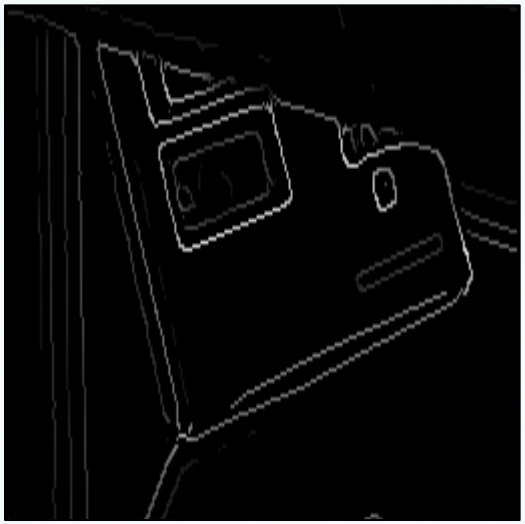
  – **src** – input image.
  – **out_lines** – output vector of lines.
  – **rho –** distance resolution of the accumulator in pixels.
  – **theta –** angle resolution of the accumulator in radians.
  – **thresh** – accumulator threshold parameter. Only those lines are returned that get enough votes (> thresh).

- Example:

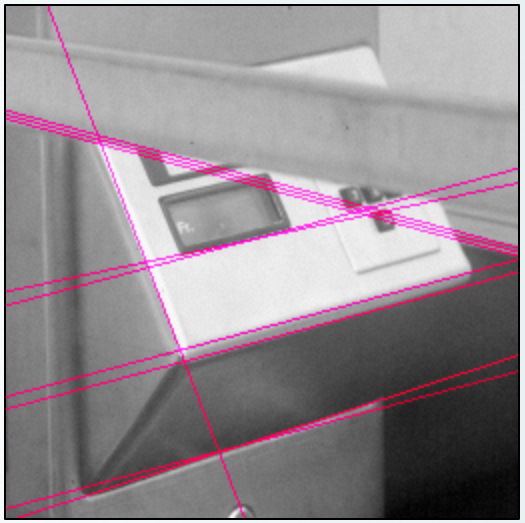
  HoughLines(src, lines, 1, CV_PI/180, 100);

Example

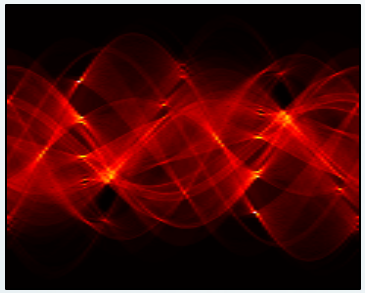
Original



Edge Detection



Found Lines



Parameter Space

Image Smoothing

- Image smoothing is used to suppress image noise or undesired image fluctuations.

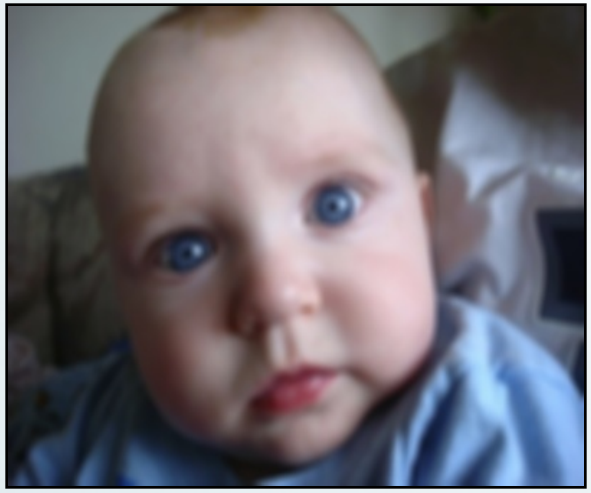
- OpenCV Fuction:
  - void **cvSmooth** (src, dst, **smooth_type**, param1, param2, param3)

  Parameters:
    - **src** – input image.
    - **dst**– output smoothed image.
    - **Smooth_type** –
      - CV_GUSSIAN
      - CV_MEDIAN
      - CV_BLUR
    - ***param1 and param2*** *- the filter window's width and height.*
    - ***param3*** *- Sigma value (only for Gaussian).*
- *Example:*

  cvSmooth(src, dst, CV_MEDIAN,3, 3);

Example

Original



medianBlur

GussianBlur

Color Spaces in OpenCV

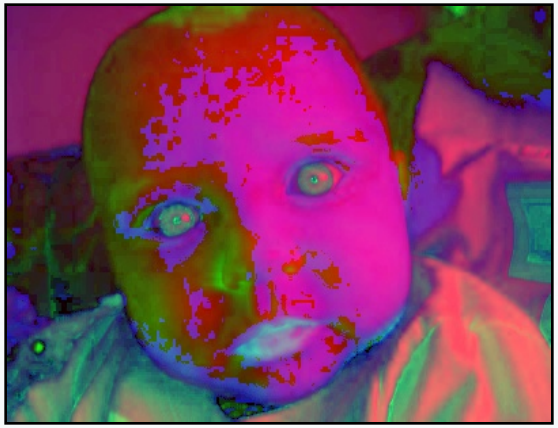
- Examples of color spaces in OpenCV:
  - **BGR (i.e., RGB)** - 3 channels color (blue, green, red).
  - **HSV** – Hue, Saturation, and Value; 3 channels.
  - **HLS** – Hue, Luminance, and Saturation; 3 channels.
  - **YCrCb or YCbCr** – Luminance, Red-difference and Blue-difference chroma components
  - **L*a*b** - Luminance dimension, a and b are the color-opponent dimensions
  - **GRAYSCALE** - Single channel.

Color Spaces in OpenCV

- OpenCV function: void **cvtColor**(**src**, **dst**, **code**)
  - Parameters:
    - **src**: input image.
    - **dst**: output image.
    - **Code:** color space conversion code (e.g., CV_BGR2GRAY, CV_BGR2HSV)
- Example:
  - cvtColor(src, bwsrc, CV_BGR2GRAY);
  - cvtColor(src, bwsrc, CV_BGR2HLS);

Example

RGB



HSV

Gray

Other Modules & Functions

- Tutorials and examples of other modules and their functions can be found in OpenCV documentation, See:
  - http://docs.opencv.org/2.4/doc/tutorials/tutorials.html
  - http://docs.opencv.org/3.0.0/modules.html
- Other useful links:
  - http://www.shervinemami.info/openCV.html
  - http://opencv-srf.blogspot.com
  - http://www.learnopencv.com
  - http://www.tutorialspoint.com/dip/

Thanks!

Please feel free to email us if you have questions?!

chihyun@mail.usf.edu
ghadh@mail.usf.edu