Computer Vision

Ch.4 Image Segmentation

Prof. Po-Yueh Chen (陳伯岳)
E-mail: pychen@cc.ncue.edu.tw
Ext: 8440
NCUE CSIE

Simple object detection by segmentation

- ✓In the previous chapter, we know there are several color models and also know how to convert between them.

Eating horse

Thresholding (1/20)

➢ What is Thresholding?

- ✓ The simplest segmentation method to define regions of interested(ROI) or objects in an image.
- ✓ This separation is based on the variation of intensity between the object pixels and the background pixels.

Thresholding (2/20)

➢ What is Thresholding?

- ✓ Once we have separated properly the important pixels, we can set them with a determined value to identify them (i.e. we can assign them a value of 0 (black), 255 (white) or any value that suits your needs).

Apple

Source: https://docs.opencv.org/3.4/Threshold_Tutorial_Theory_Example.jpg

Thresholding (3/20)

➢ The thresholding types

✓OpenCV offers the function cv::threshold to perform thresholding operations.

✓We can effectuate 5 types of Thresholding operations with this function.

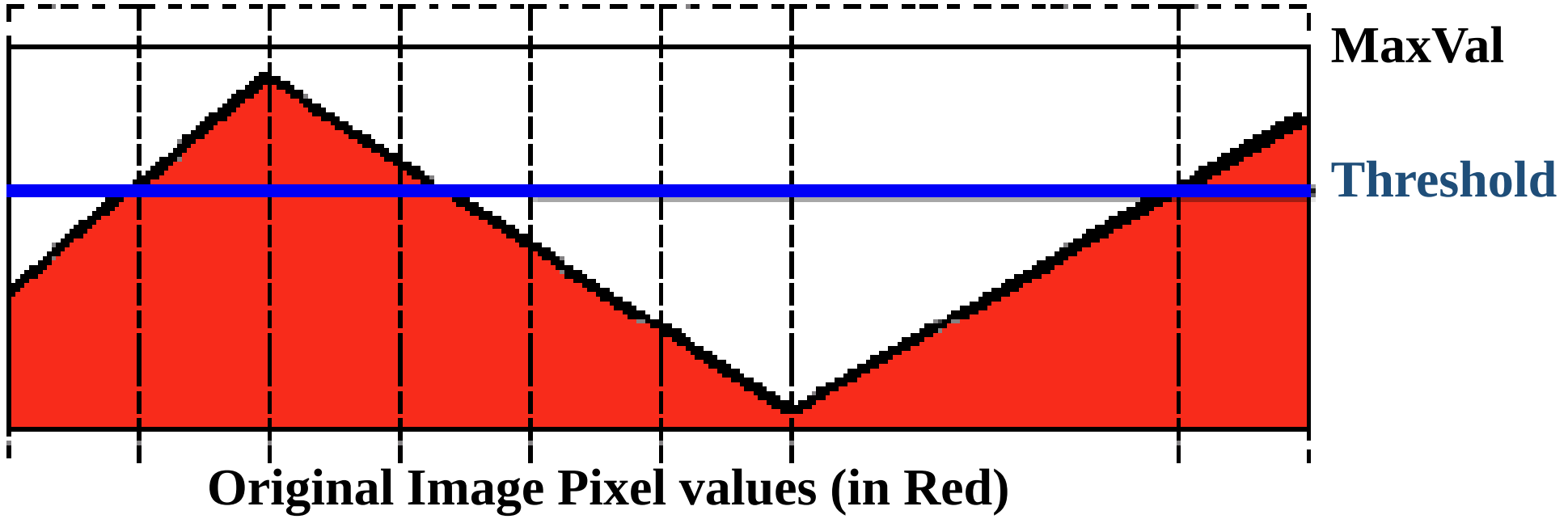
Types:

1. **THRESH_BINARY**
2. **THRESH_BINARY_INV**
3. **THRESH_TRUNC**
4. **THRESH_TOZERO**
5. **THRESH_TOZERO_INV**

Thresholding (4/20)

➢ The thresholding types

✓ To illustrate how these thresholding processes work, let's consider that we have a source image with pixels with intensity values $src(x, y)$. The plot below depicts this.



✓ The horizontal blue line represents the threshold(fixed).

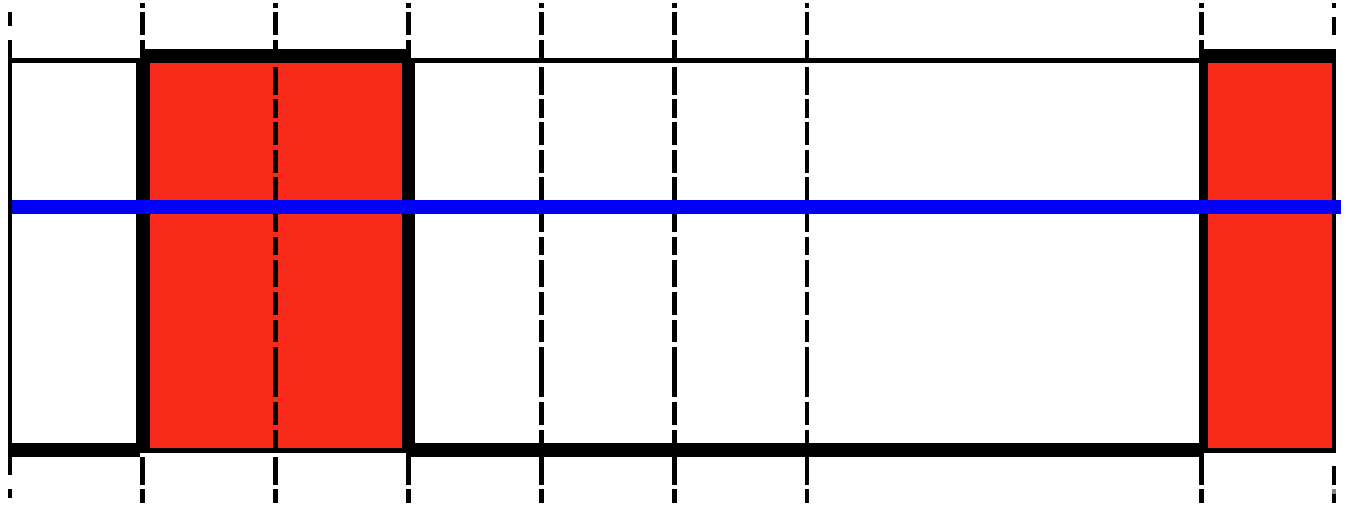
Thresholding (5/20)

➢ Threshold Binary

1. THRESH_BINARY

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

✓ Binarization



THRESH_BINARY

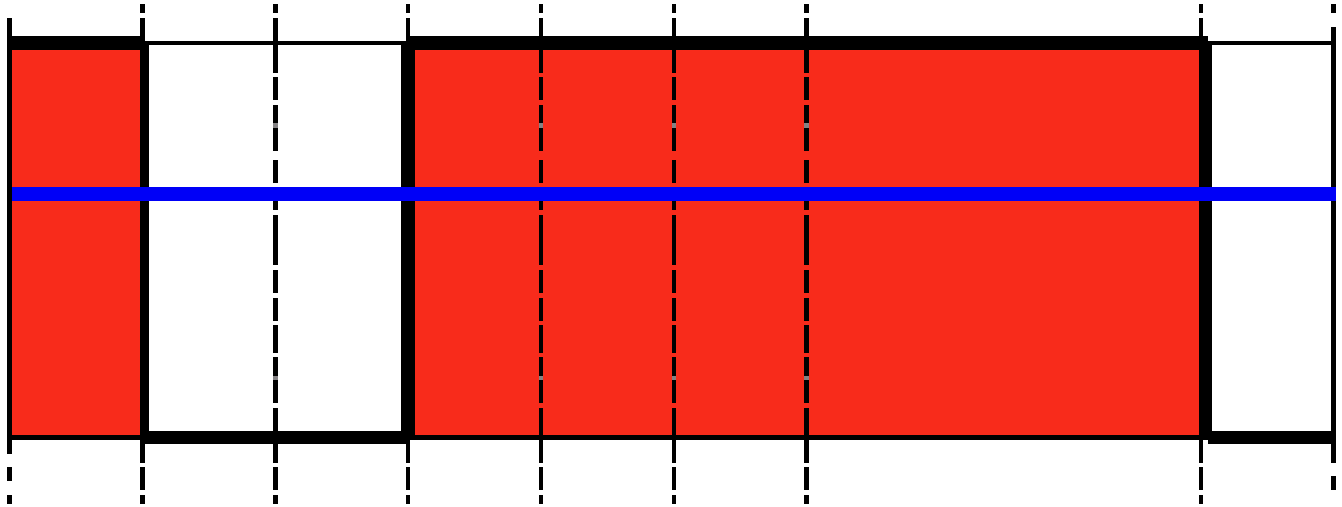
Thresholding (6/20)

➢ Threshold Binary, Inverted

2. THRESH_BINARY_INV

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

- Higher values will be set to zero, and lower values will be set to the max value.



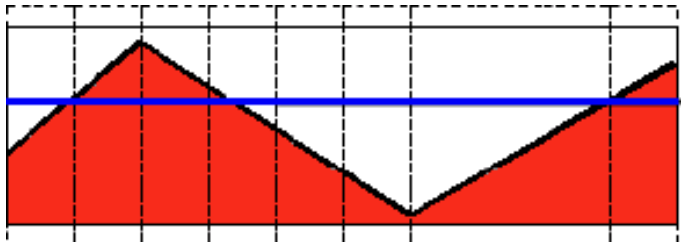
THRESH_BINARY_INV

Thresholding (7/20)

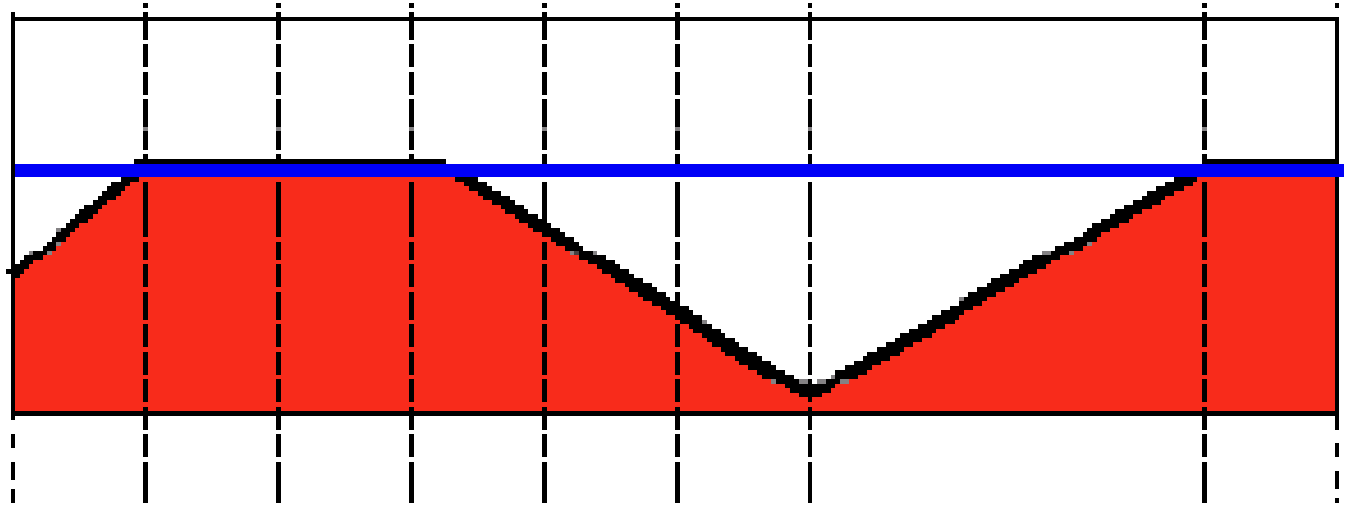
➢ Truncate

3. THRESH_TRUNC

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



Original Image



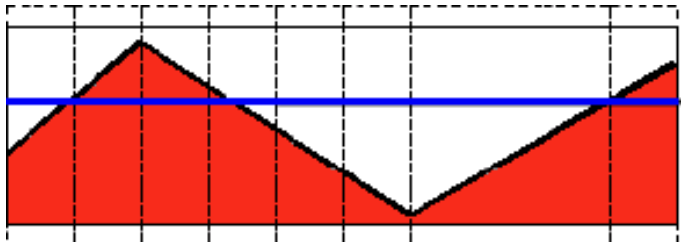
THRESH_TRUNC

Thresholding (8/20)

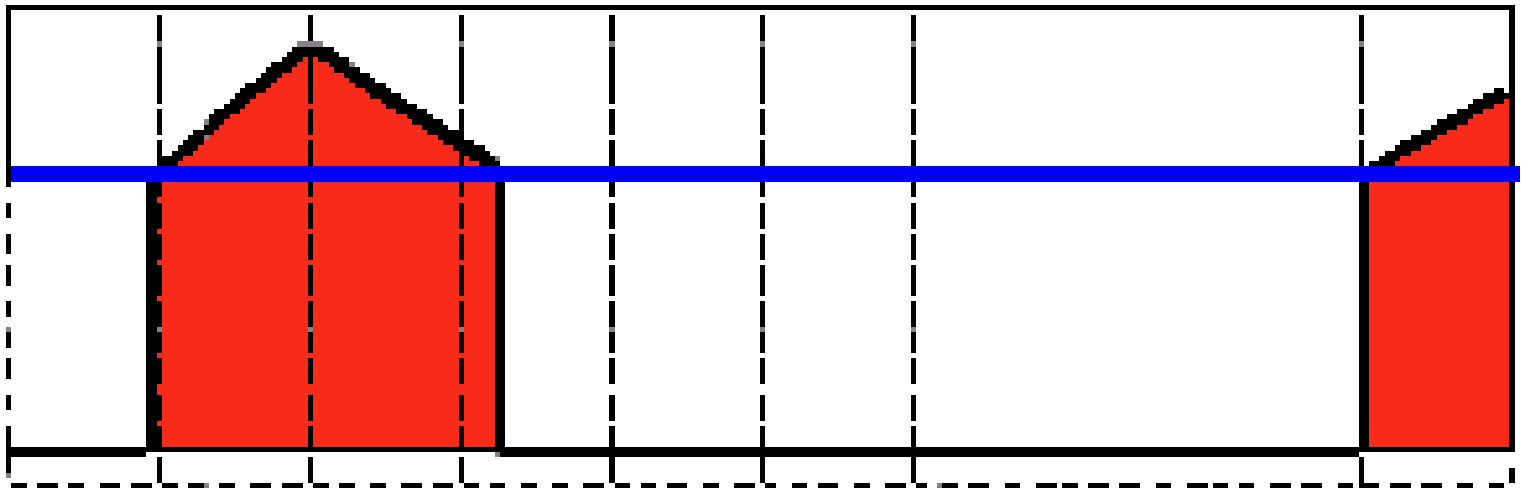
➢ Threshold to Zero

4. THRESH_TOZERO

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



Original Image



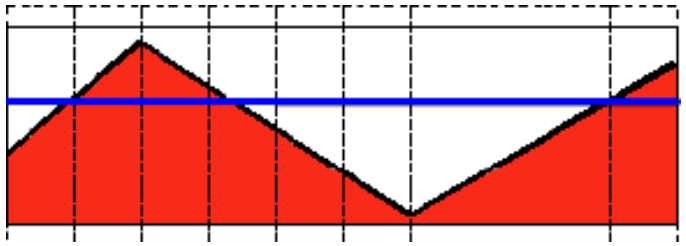
THRESH_TOZERO

Thresholding (9/20)

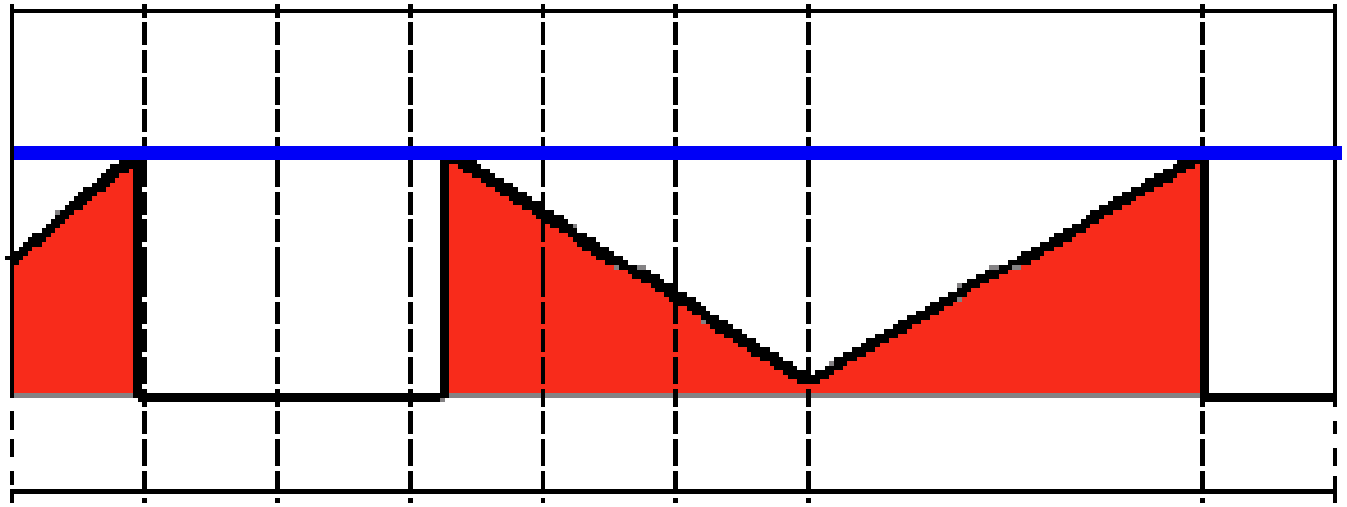
➢ Threshold to Zero, Inverted

5. THRESH_TOZERO_INV

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



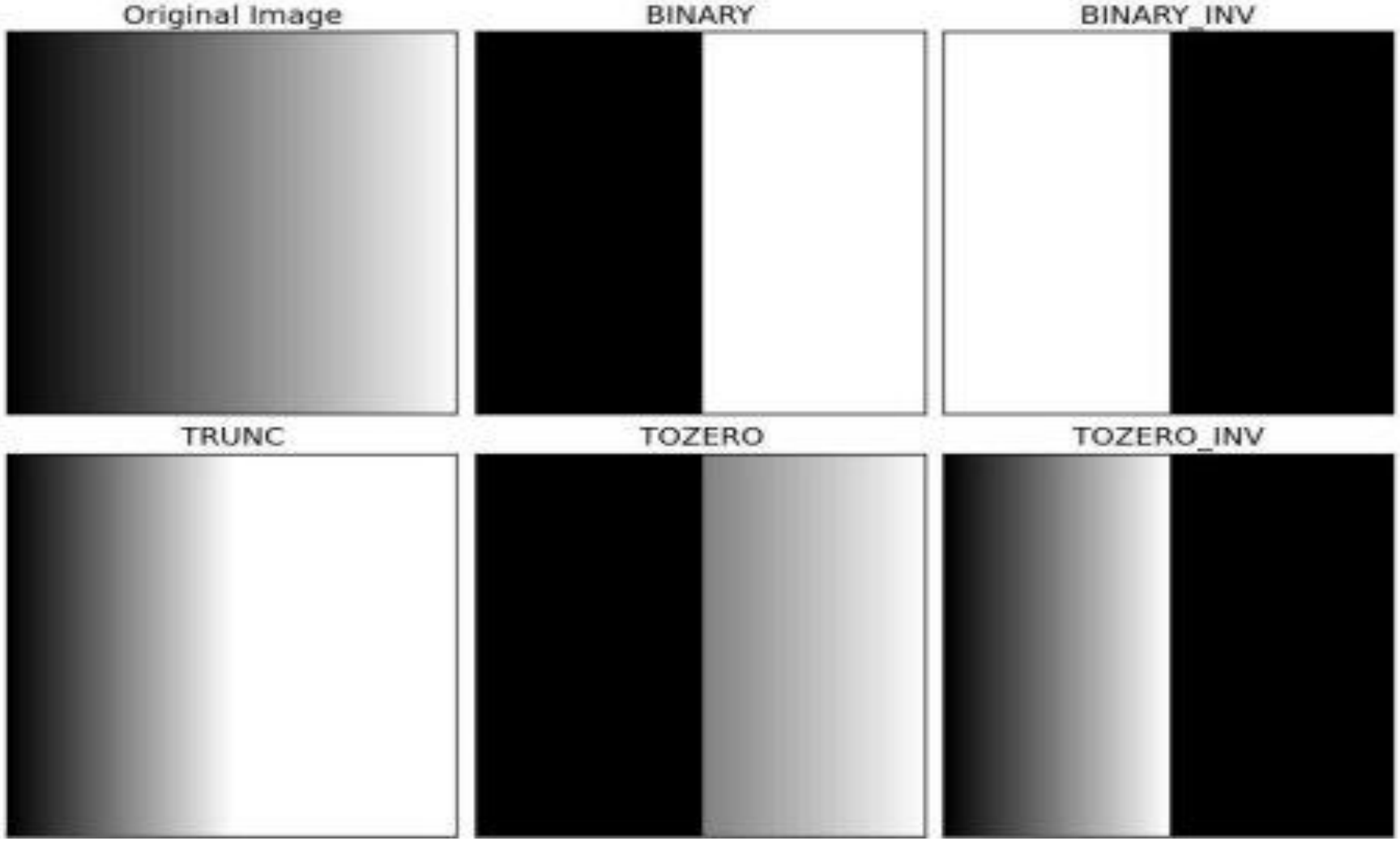
Original Image



THRESH_TOZERO_INV

Thresholding (10/20)

- **The yields of thresholding example**



Source: https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html

Thresholding (11/20)

✓ To extract High-Intensity Pixels

- Sometimes, how to set a good threshold is a big issue in computer vision.
- Let us try on two different threshold values.

Original Image - Lena

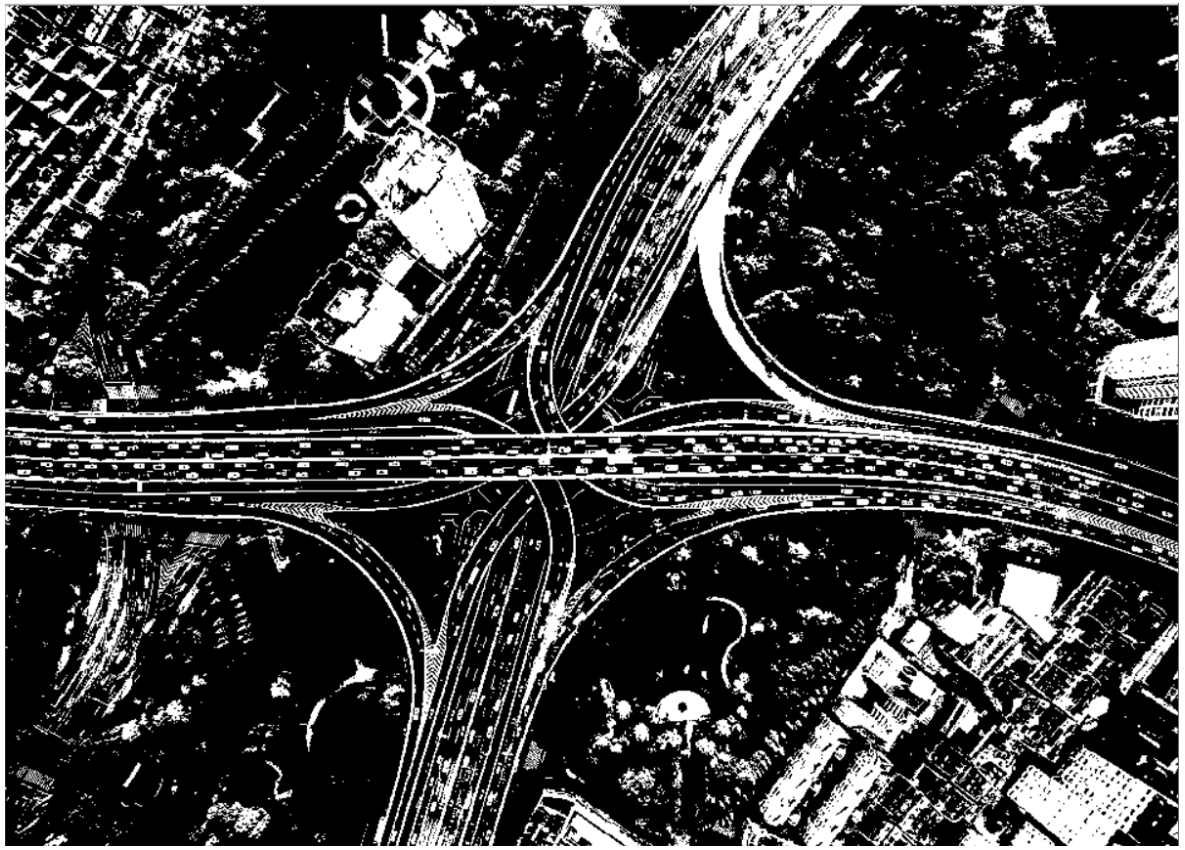
THRESH_BINARY threshold = 100

THRESH_BINARY threshold = 135

Thresholding (12/20)

- **Application in real life.**

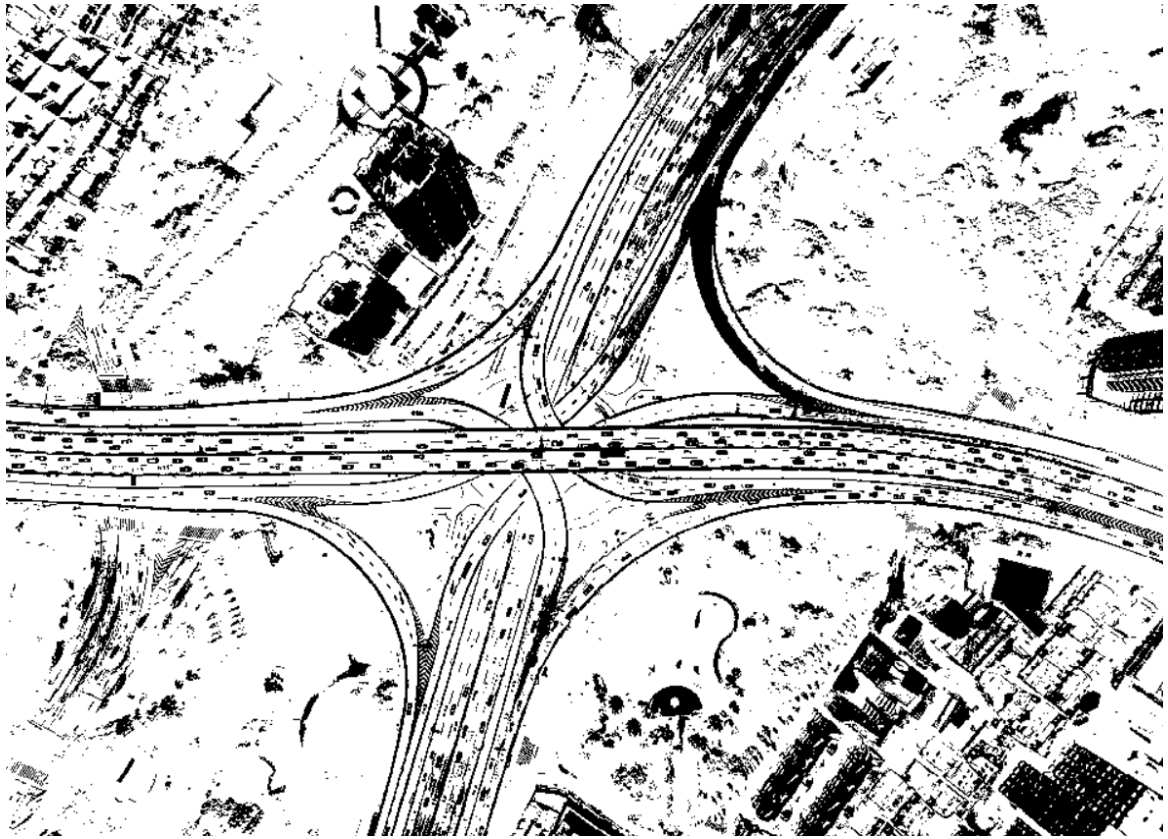
Original Image



THRESH_BINARY

Thresholding (13/20)

- **Application in real life.**



THRESH_BINARY_INV

THRESH_TOZERO

Thresholding (14/20)

➢ Application in real life.

Original Image

The thresholding result

- The example of lane marking detection.

Source: Computer Vision - Color by Chen Hua-Tsung

Thresholding (15/20)

➢ Application in real life.

- **There are some noise problem with vehicle sensor.**

- 衝突被害軽減ブレーキ　豪雨で検証

Thresholding (16/20)

➢ Code

Syntax:

threshold(src, dst, double thresh, double maxval, int type);

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.

Ex.

1. **THRESH_BINARY**
2. **THRESH_BINARY_INV**
3. **THRESH_TRUNC**
4. **THRESH_TOZERO**
5. **THRESH_TOZERO_INV**

Thresholding (17/20)

- **Demo code**

  The header as following in below…

```
#include "opencv2/core/utility.hpp“

#include "opencv2/imgproc.hpp"
#include "opencv2/imgcodecs.hpp"

#include "opencv2/highgui.hpp"
```

Thresholding (18/20)

- **Demo code**

```
const char* trackbar_type = "Type: \n 0: Binary \n 1: Binary Inverted \n 2: Truncate \n 3: To Zero \n 4: To Zero Inverted";
```

First, Declare the threshold parameters.

```
int threshold_value = 0;
int threshold_type = 3;        //Default Value Start on 4. Threshold_To_Zero
int const max_value = 255;
int const max_type = 4;
int const max_binary_value = 255;
Mat src, src_gray, dst;

static void Threshold_Demo(int, void*)
{
   threshold(src_gray, dst, threshold_value, max_binary_value, threshold_type);
   imshow(window_name, dst);
}
```

Thresholding (19/20)

➢ Demo code

A part of demo code…

```
int main(int argc, char** argv)
{
    String imageName("lena.jpg"); // by default root file path.
    if (argc > 1)
    {
        imageName = argv[1];
    }
    src = imread(samples::findFile(imageName), IMREAD_COLOR); // Load an image form defualt root file.
    if (src.empty())
    {
        cout << "Cannot read the image: " << imageName << std::endl;
        return -1;
    }
    cvtColor(src, src_gray, COLOR_BGR2GRAY); // Convert the image to Gray
    namedWindow(window_name,WINDOW_GUI_NORMAL); // Create a window to display results
    createTrackbar(trackbar_type, window_name, &threshold_type,max_type, Threshold_Demo);
    // Create a Trackbar to choose type of Threshold
    createTrackbar(trackbar_value, window_name, &threshold_value, max_value, Threshold_Demo);
    // Create a Trackbar to choose Threshold value
    Threshold_Demo(0, 0); // Call the function to initialize

    waitKey();
    destroyAllWindows();
    return 0;
}
```

Thresholding (20/20)

➢ Demo code(Major part)

```
const char* window_name = "Threshold Demo";
const char* trackbar_type = "Type: \n 0: Binary
\n 1: Binary Inverted \n 2: Truncate \n 3: To Zero
\n 4: To Zero Inverted";
const char* trackbar_value = "Value";
cvtColor(src, src_gray, COLOR_BGR2GRAY);
// Convert the image to Gray
namedWindow(window_name,WINDOW_GUI_NORMAL);
// Create a window to display results

createTrackbar(trackbar_type, window_name, &threshold_type,max_type, Threshold_Demo);
// Create a Trackbar to choose type of Threshold
createTrackbar(trackbar_value, window_name, &threshold_value, max_value, Threshold_Demo);
// Create a Trackbar to choose Threshold value

Threshold_Demo(0, 0); // Call the function to initialize
```

Demo

Practice

✓ **Free images library: Pixabay, Librestock**

Exercise #1

Find some objects in any image for using 5 thresholding method.

✓ **Note: (Full Project file)**

.sln, *.exe ,*.ppt(or *.pptx) and ***original image*** are necessary.

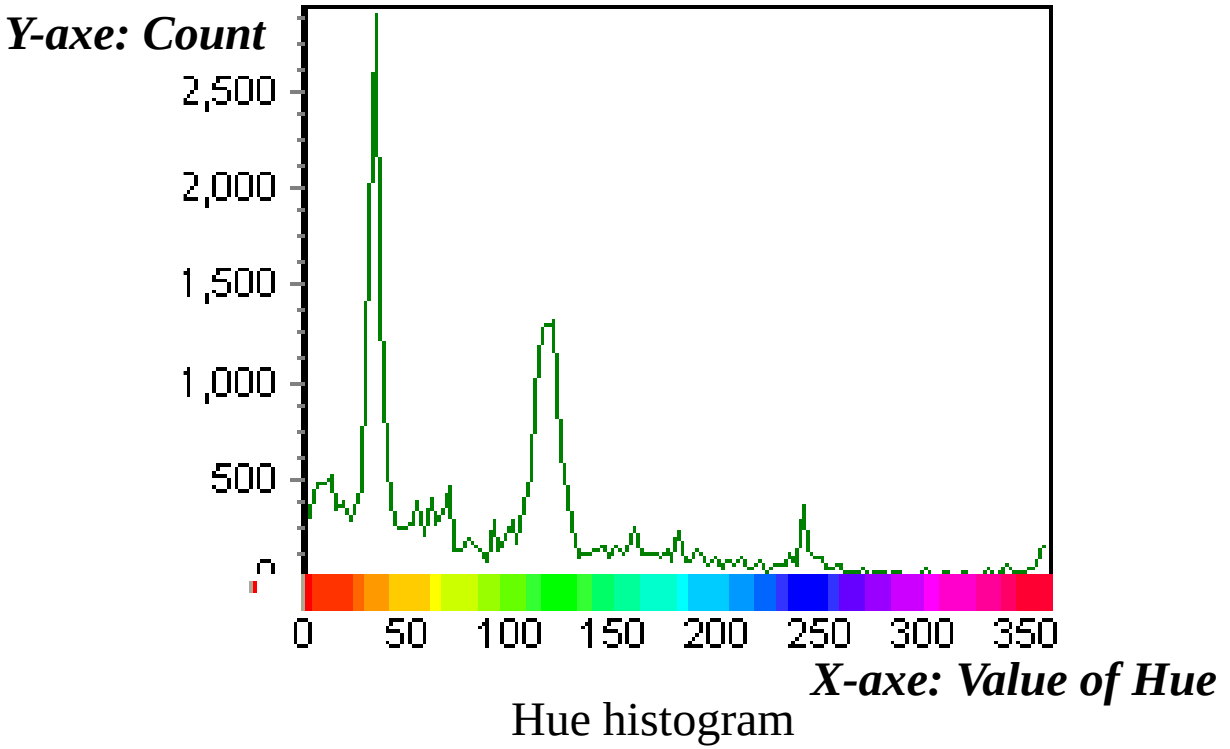
Color-based Segmentation (1/12)

In some kinds of videos, there are dominant colors which take a large proportion in the frames.

- ✓ Sports videos: Field colors or court colors
- ✓ Traffic videos: Pavement color

Color-based Segmentation (2/12)

How to extract the dominant color?



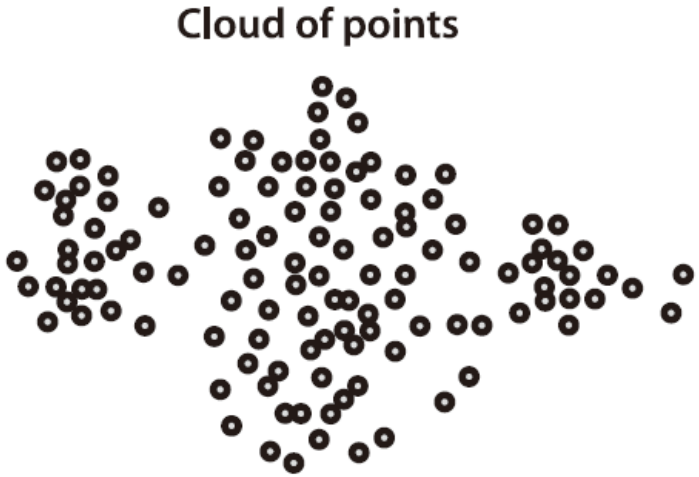
Hue histogram

- A color histogram is a representation of the distribution of colors in an image.

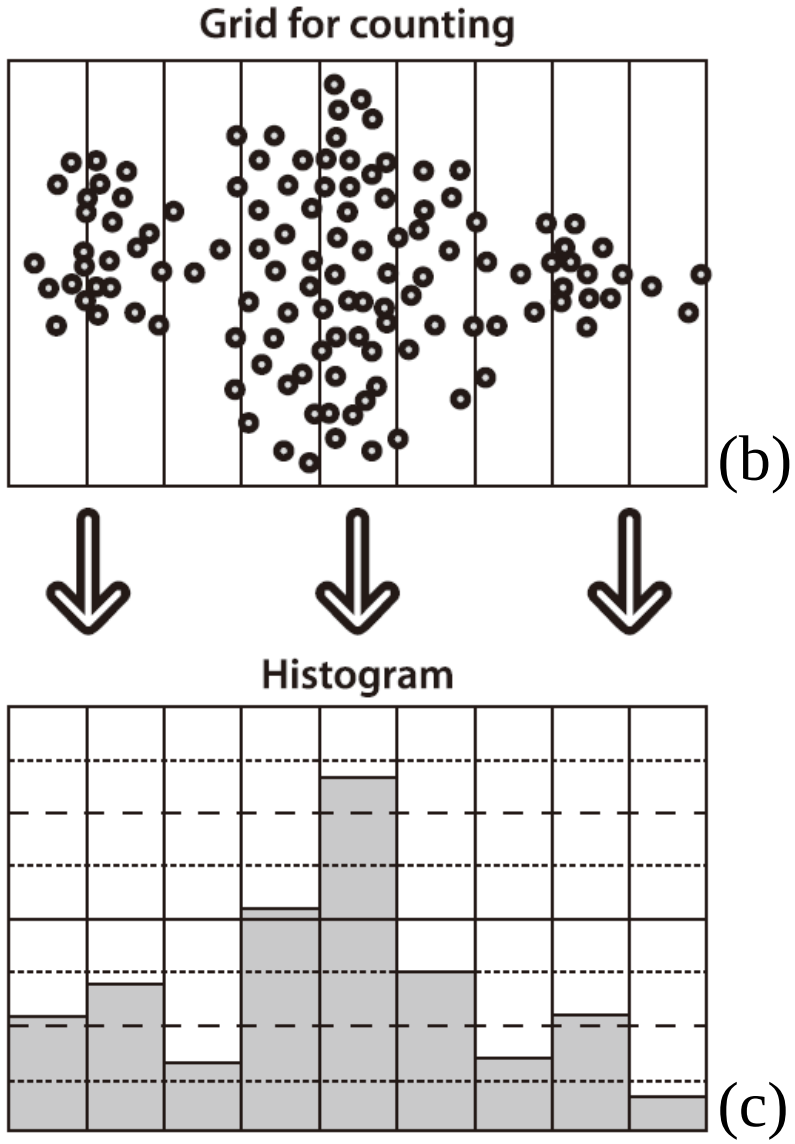
Source: Computer Vision - Color by Chen Hua-Tsung

Color-based Segmentation (3/12)

➢ Histogram



(a)



- Start with a cloud of points (fig. a).
- A counting grid is imposed (fig. b) that yields a one-dimensional histogram of point counts (fig. c).

➢ The histogram shows the point count of each grid cell.

Source: Computer Vision - Color by Chen Hua-Tsung

Color-based Segmentation (4/12)

➢ Histogram

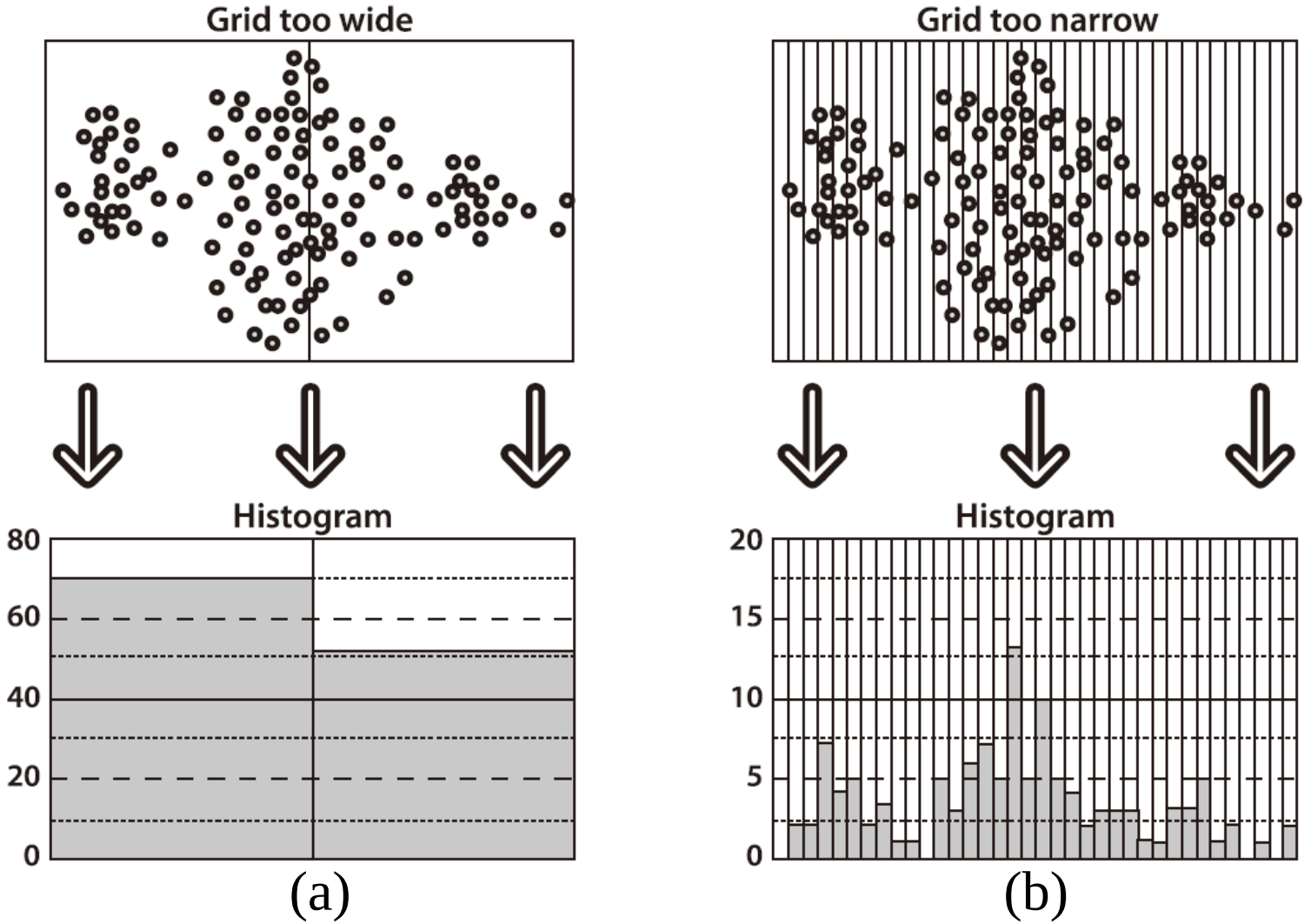
A histogram's accuracy highly depends on its grid size:

Fig. a.

A grid that is too wide will yield too much spatial averaging in the histogram counts.

Fig. b.

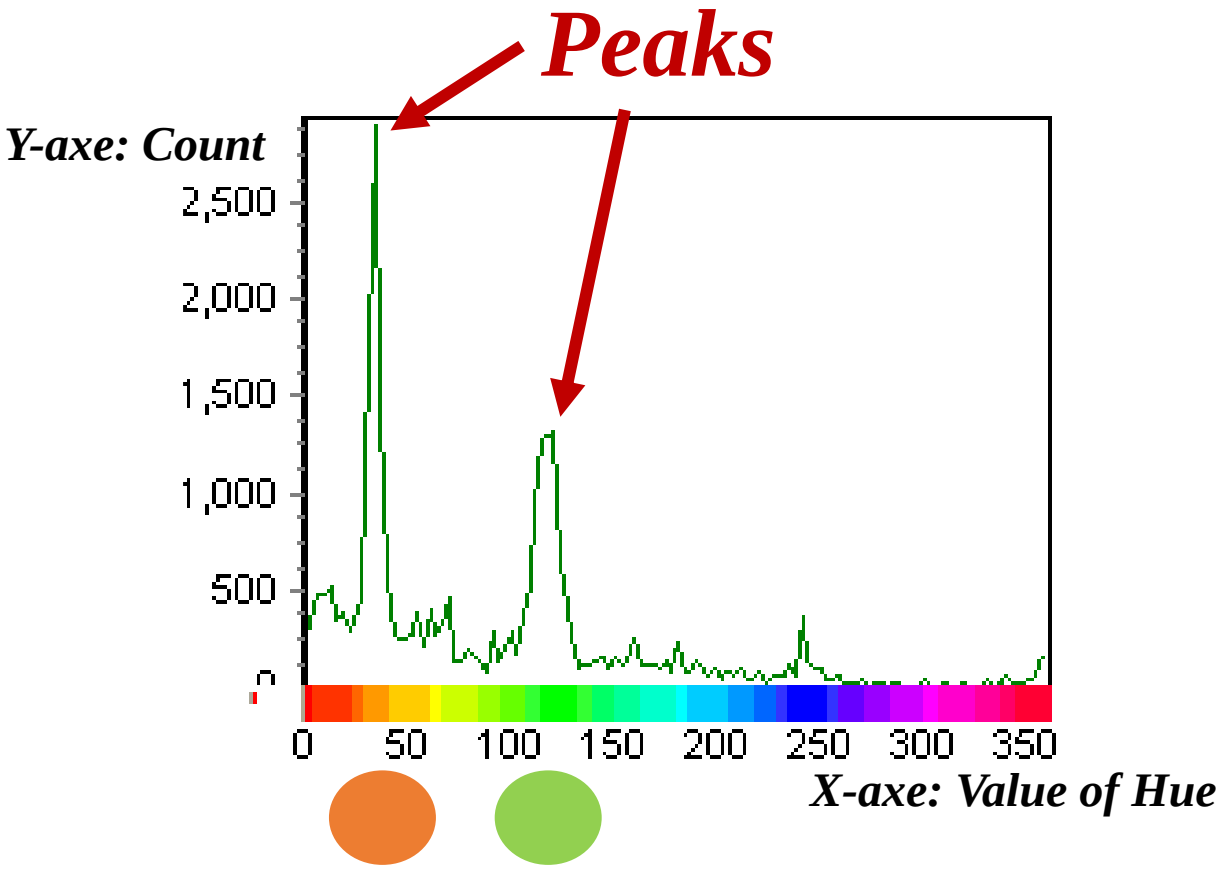
A grid that is too small will yield "spiky" and singleton results from too little averaging.



Source: Computer Vision - Color by Chen Hua-Tsung

Color-based Segmentation (5/12)

➢ Histogram

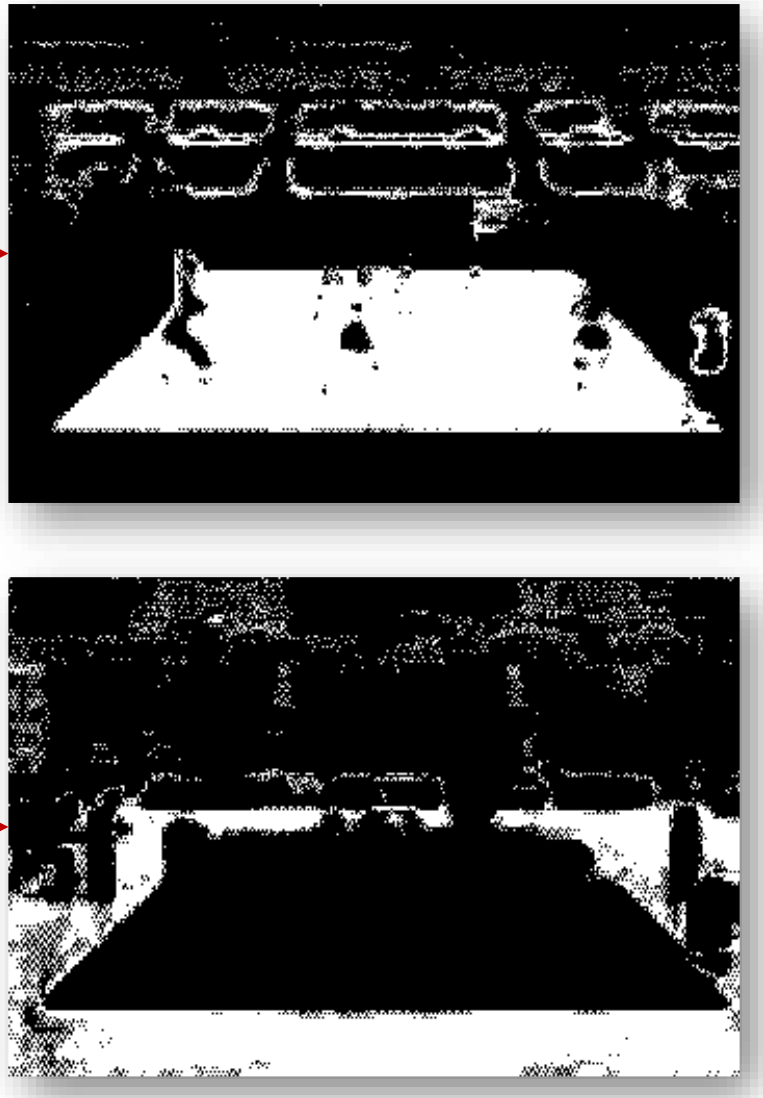


- For example, two dominant colors can be found by the peaks of the histogram.

Source: Computer Vision - Color by Chen Hua-Tsung

Color-based Segmentation (6/12)

➢ Histogram

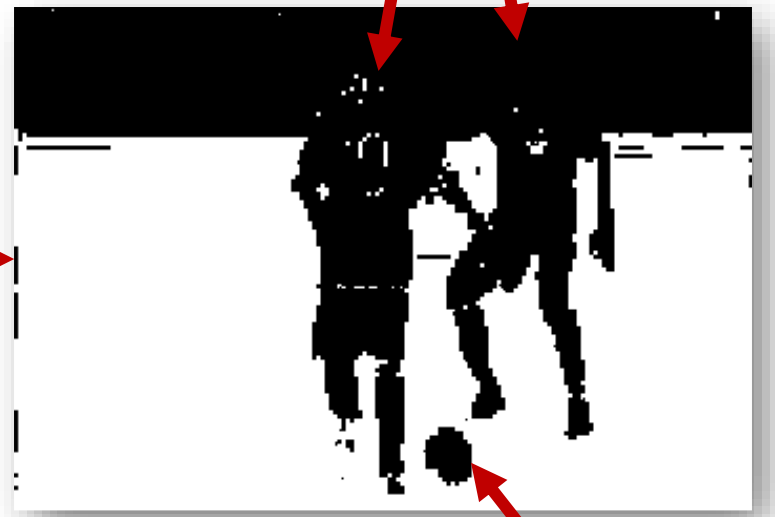


✓ Non-dominant color regions can be taken as foreground candidates.

Source: Computer Vision - Color by Chen Hua-Tsung

Color-based Segmentation (7/12)

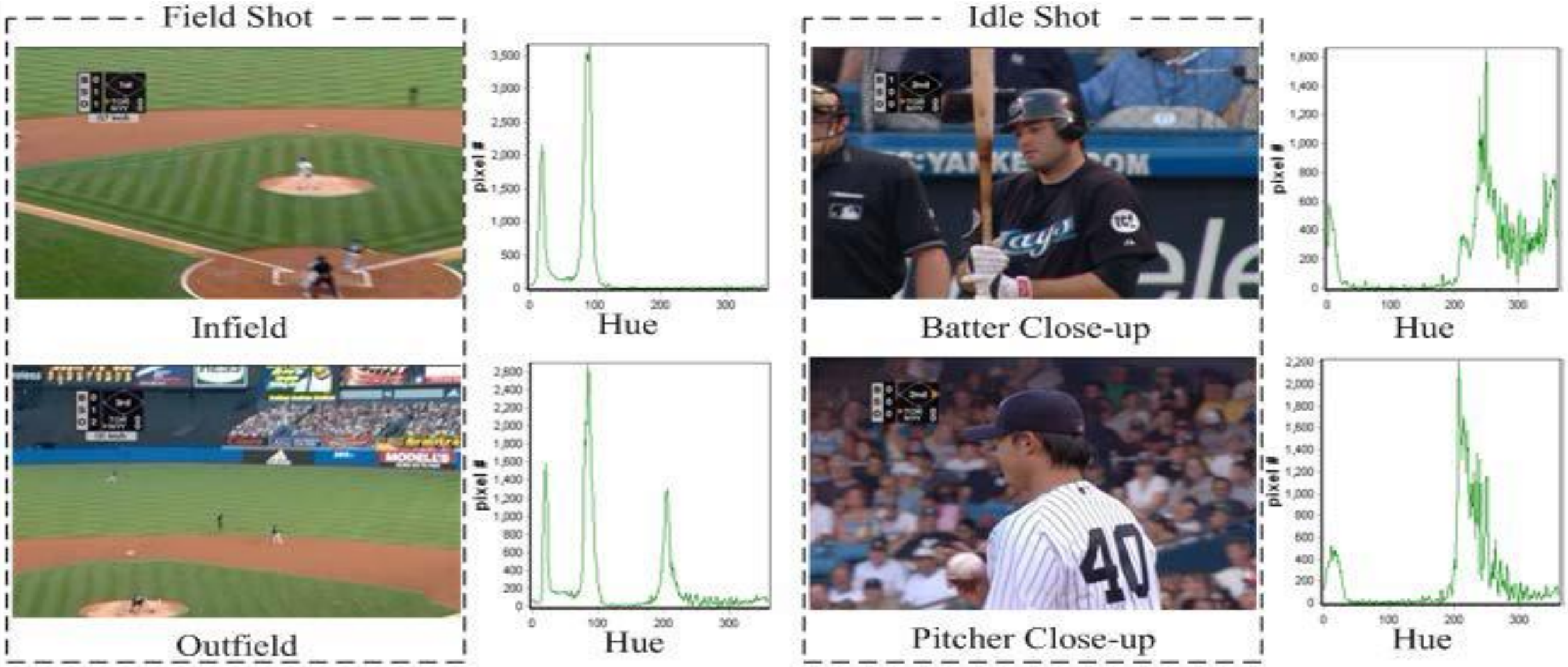
➢ Object extraction



- ✓ The field region can be obtained by the dominant color.
- ✓ Remove it then we can see the objects extracted.

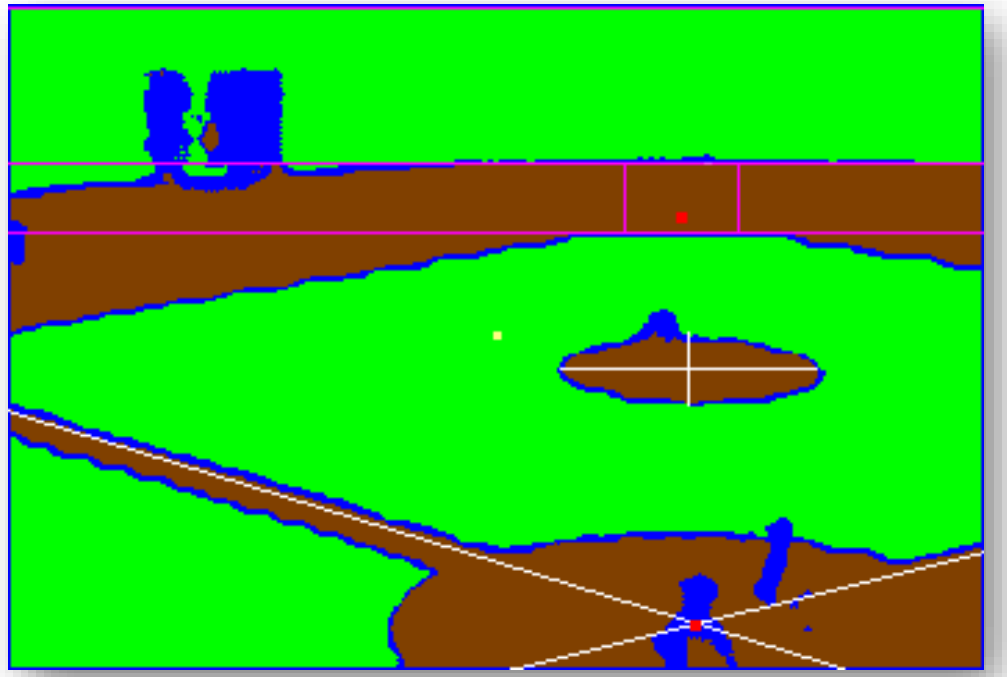
Color-based Segmentation (8/12)

- **Scene classification**



Color-based Segmentation (9/12)

- **Scene classification**



- Extract dominant colors by hue histogram.
- Segment the scene of the image into soil and grass regions.

Color-based Segmentation (10/12)

- **Mean shift**

The function implements the filtering stage of meanshift segmentation, that is, the output of the function is the filtered "posterized" image with color gradients and fine-grain texture flattened.

Color-based Segmentation (11/12)

➢ Mean shift

At every pixel (X,Y) of the input image (or down-sized input image, see below) the function executes meanshift iterations, that is, the pixel (X,Y) neighborhood in the joint space-color hyperspace is considered:

$$(x, y) : X - \mathtt{sp} \leq x \leq X + \mathtt{sp},$$

$$Y - \mathtt{sp} \leq y \leq Y + \mathtt{sp},$$

$$||(R, G, B) - (r, g, b)|| \leq \mathtt{sr}$$

Color-based Segmentation (12/12)

➢ Code

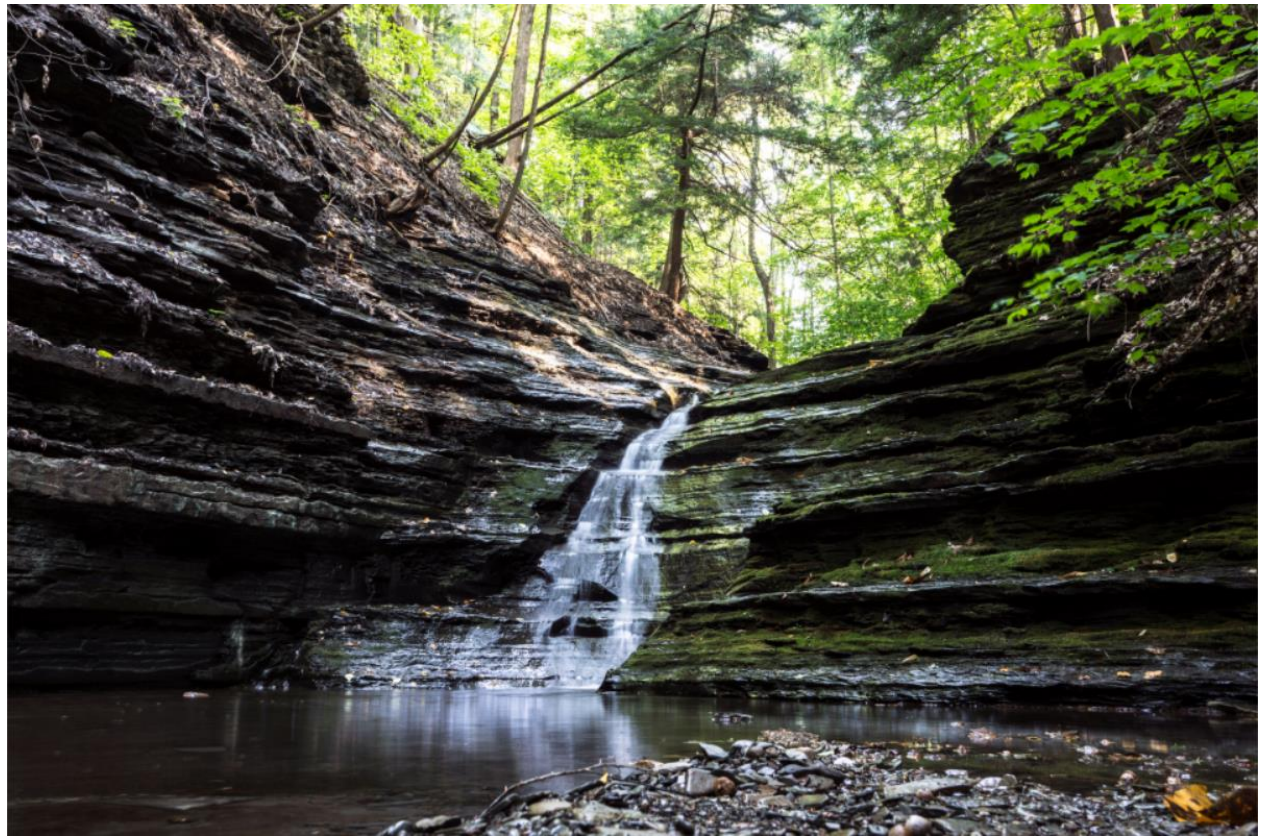
Syntax:

pyrMeanShiftFiltering(src, dst, sp, sr, int maxLevel, TermCriteria);

src – input array (single-channel, 8-bit or 32-bit floating point).
dst – The destination image of the same format and the same size as the source
sp – The spatial window radius.
sr – The color window radius.
maxLevel – Maximum level of the pyramid for the segmentation.
termcrit – Termination criteria: when to stop mean shift iterations.

Demo

Demo



Original Image

Result Image

pyrMeanShiftFiltering(src, dst, 30, 60, 3);

Practice

Thanks!

Any questions?