Introduction to OpenCV

By Noriaki Kono

Computer Vision (CV)

Why CV?

Pros:
- Needs very few images to “learn”
- All results are expected and understood

Cons:
- Generalized solution can complicate the model
- Rigid and inflexible model
  - Requires new solutions for new situations
- Can be very difficult to achieve optimal results

Image Formation

What is depth?

- Images are 2D, implying no depth
- Human brains assume depth
  - Object overlap
  - Shadows

Image Formation

What are edges?

- Two very basic types of edges:
  - Hard Edges - Edges which transition quickly
  - Soft Edges - Edges which transition slowly
- Derivatives help measure edges

Image Formation

What are edges?

Hard Edge Soft Edge

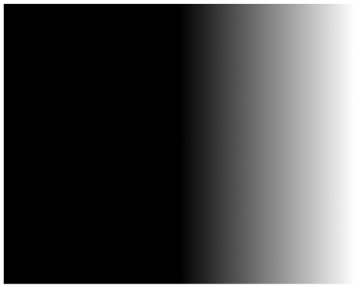


Image Formation

What are edges?

How many edges?

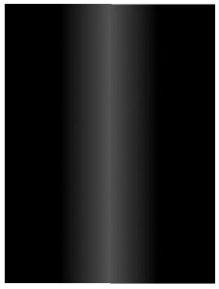


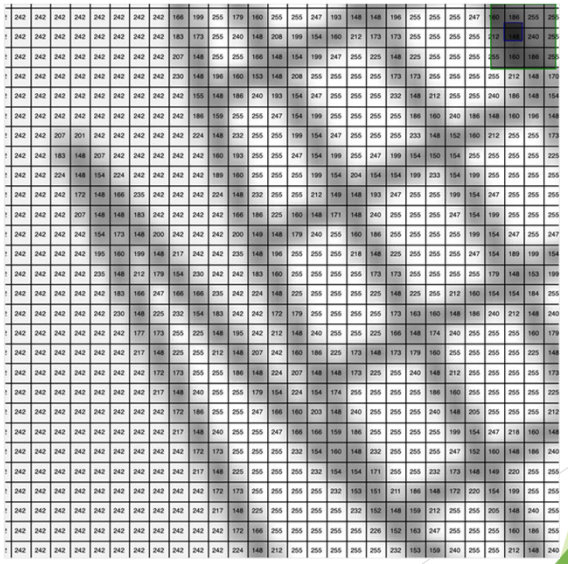
Image Formation

What is a pixel?

- Measures Intensity
  - Information carrier
- A byte translates to 256 values (0-255)
- 0 - represents black
- 255 - represents white

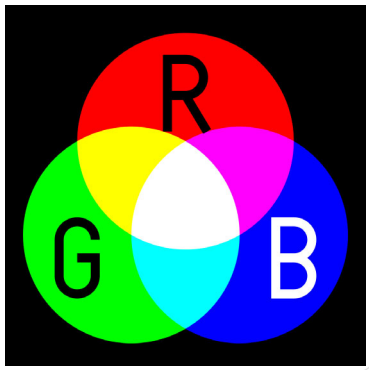
Image Formation

What is a pixel?



Color Space - RGB

- RGB color images are created using 3 grayscale images
- Red, green, and blue color channel values range from 0 – 255 due to 8-bit restrictions
- RGB produces colors using additive color mixing



https://en.wikipedia.org/wiki/RGB_color_model#/media/File:AdditiveColor.svg

Image Formation

What is a pixel?

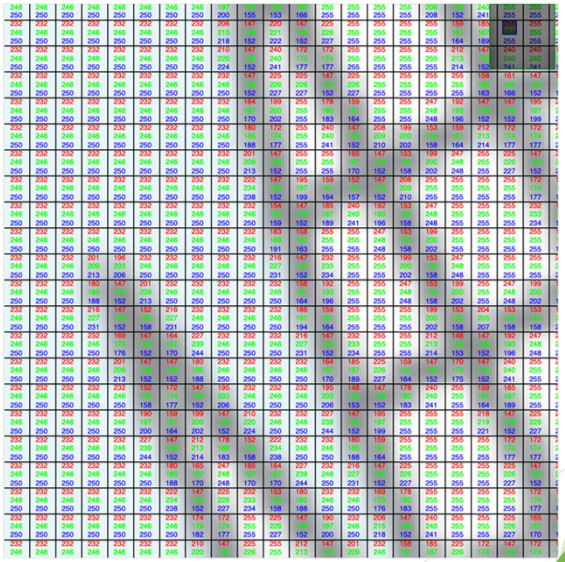
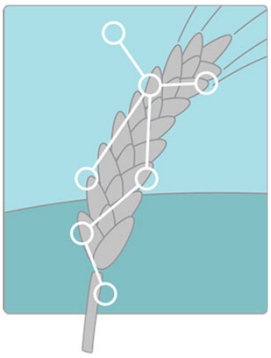


Image Formation

Color Space

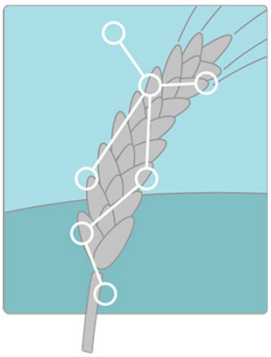


RGB - Red, Green, Blue color channels

- Uses 3 images to construct color
- Each image is created with intensity values (grayscale)
- Order for each image matters
  - Computer automatically assumed RGB order of images

Image Formation

Color Space



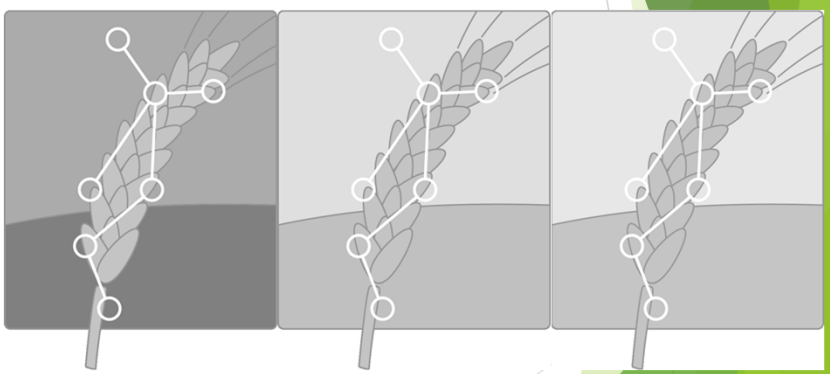
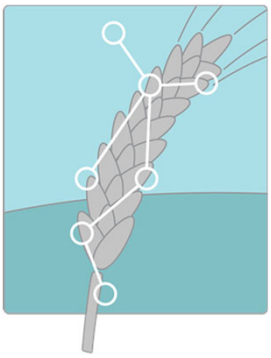
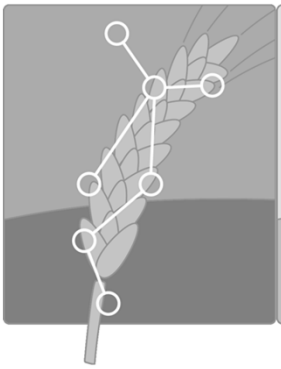


Image Formation

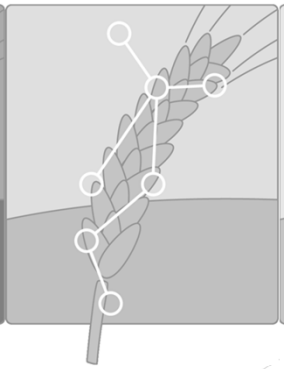
Color Space



Red



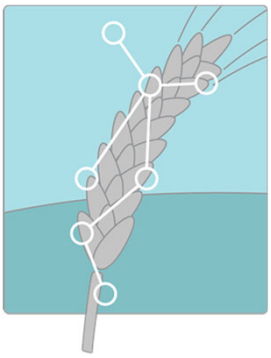
Green



Blue

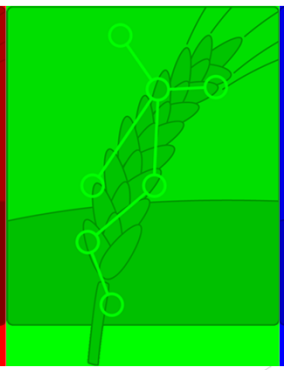
Image Formation

Color Space



Red

Green



Blue

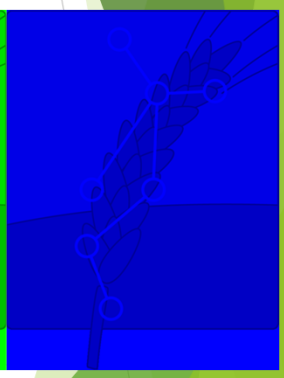
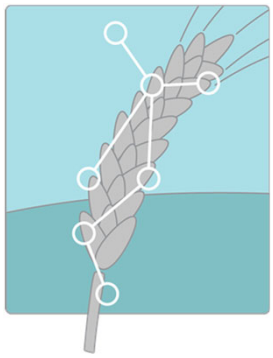


Image Formation

Color Space



Images combined in RGB order

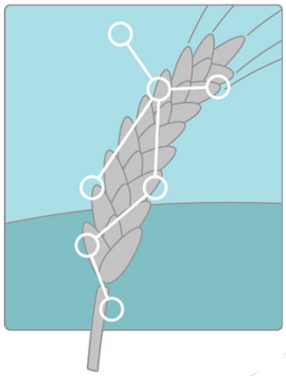
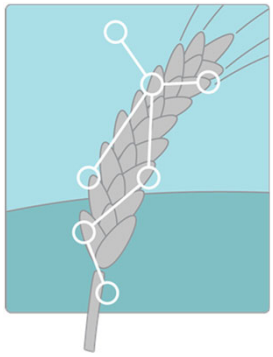
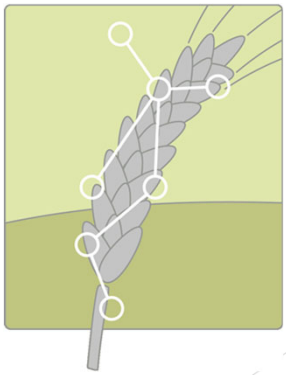


Image Formation

Color Space



Images combined in GBR order



Color Space - RGB

- Most commonly used color space
- OpenCV uses BGR instead of the traditional RGB order

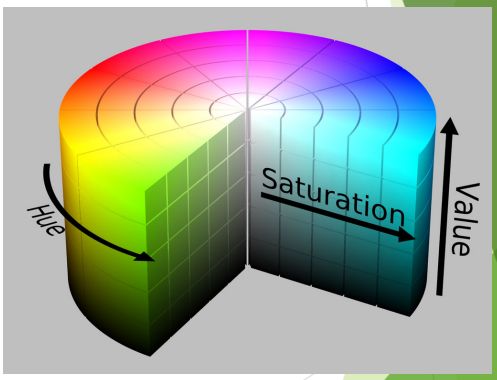
- The “distance” between colors is very large and hard to group
- Below example shows the limitation of analyzing RGB color space:

(0, 0, 255) (128, 0, 255) (128, 255, 255) (56, 0, 255) (255, 255, 255)

Color Space - HSV

	Real World	OpenCV (8-bits)
H - Hue:	0 - 359 degrees	0 - 179
S - Saturation:	0 - 100%	0 - 255
V - Value:	0 - 100%	0 - 255

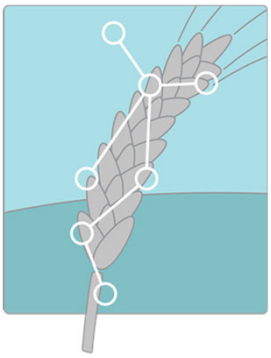
- Hue is the primary color information
- Saturation represents the amount of gray
- Value represents the intensity of the color



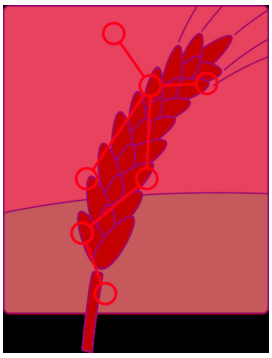
https://en.wikipedia.org/wiki/HSL_and_HSV#/media/File:HSV_color_solid_cylinder_saturation_gray.png

Color Space

RGB

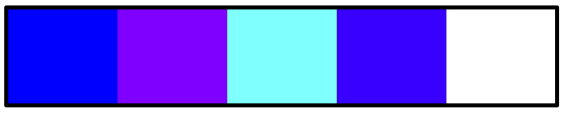


HSV



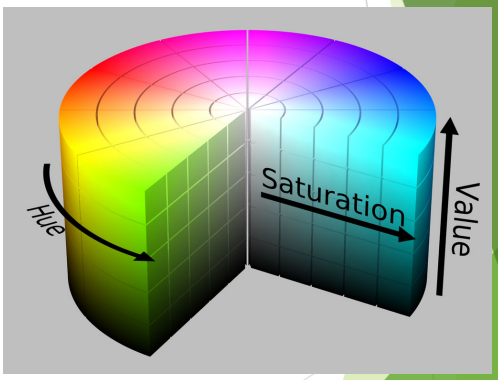
Color Space - HSV

RGB



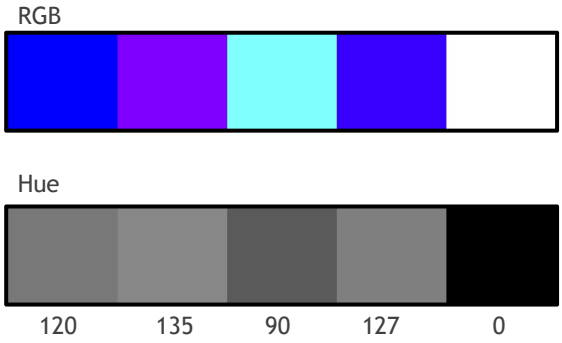
HSV

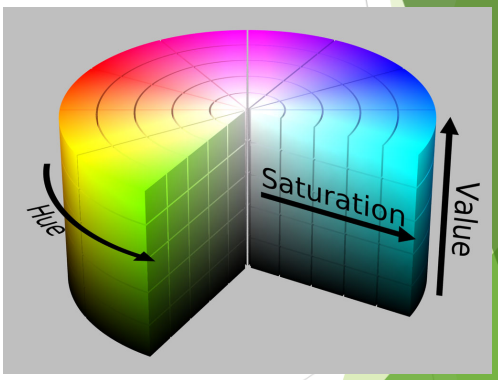
Hue



https://en.wikipedia.org/wiki/HSL_and_HSV#/media/File:HSV_color_solid_cylinder_saturation_gray.png

Color Space - HSV



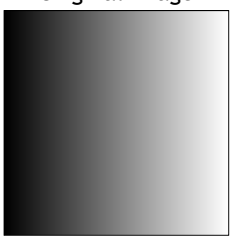


https://en.wikipedia.org/wiki/HSL_and_HSV#/media/File:HSV_color_solid_cylinder_saturation_gray.png

Image Thresholding - Simple Thresholding

- If pixel value is greater than a threshold value, it is assigned one value (usually 255, white), else it is assigned to 0 (black).
- Image is transformed into a form of a binary image.

Original Image



Threshold Image

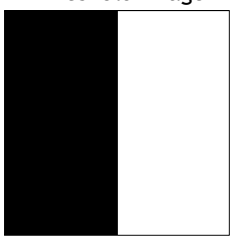
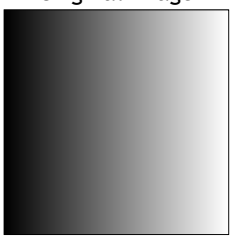


Image Thresholding - inRange Thresholding

- If pixel value is in between the lower and upper threshold values, it is assigned 255 (white), else it is assigned the value 0 (black).
- Image is transformed into a form of a binary image.

Original Image



inRange Image

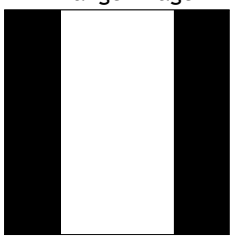
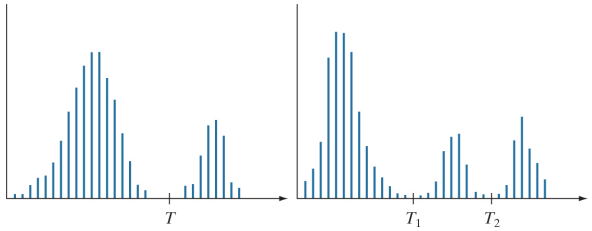


Image Thresholding

- When T is a constant applicable over an entire image, the process given in the equation is referred to as *global thresholding*.
- When the value of T changes over an image, we use the term *variable thresholding*. The terms local or regional thresholding are used sometimes to denote variable thresholding in which the value of T at any point (x, y) in an image *depends on properties of a neighborhood of* (x, y) (for example, the average intensity of the pixels in the neighborhood).
- If $\boldsymbol{T}$ *depends on the spatial coordinates* $\boldsymbol{(x, y)}$ *themselves*, then variable thresholding is often referred to as *dynamic or adaptive thresholding*.

Image Thresholding

a b



- Suppose that the intensity histogram corresponds to an image, $f(x,y)$, composed of light objects on a dark background, in such a way that object and background pixels have intensity values grouped into two dominant modes. One obvious way to extract the objects from the background is to select a threshold, T, that separates these modes.

$$g(x,y) = \begin{cases} 1 & if\ f(x,y) > T \\ 0 & if\ f(x,y) \le T \end{cases} \qquad g(x,y) = \begin{cases} a & if\ f(x,y) > T_2 \\ b & if\ T_1 < f(x,y) \le T_2 \\ c & if\ f(x,y) < T_1 \end{cases}$$

Image Thresholding - Adaptive Thresholding

- In simple thresholding, a global value is used as threshold value. But it may not be good in all the conditions where image has different lighting conditions in different areas. In that case, we use *adaptive thresholding in which the algorithm calculates the threshold for a small regions of the image*. So we get different thresholds for different regions of the same image and it gives us better results for images with varying illumination.

- Adaptive Method - Computationally decides how the thresholding value is calculated.
  - Mean – Threshold is determined by the mean of the neighborhood area.
  - Gaussian – Threshold is determined by the weighted sum of the neighborhood values. The weights are determined by a gaussian window.

Image Thresholding - Adaptive Thresholding

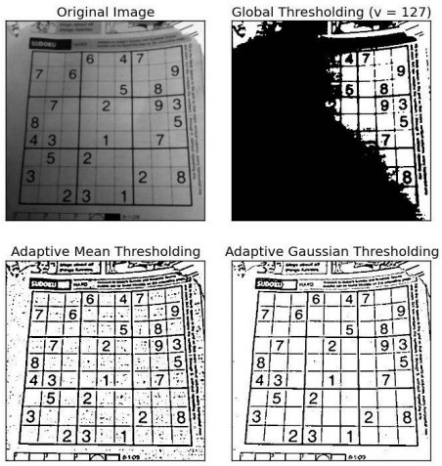


Image Thresholding - Otsu Thresholding

- In global thresholding, we used an arbitrary value for threshold value. So, how can we know a value we selected is good or not?
- Answer is, *trial and error* method.
- But consider a bimodal image (*In simple words, bimodal image is an image whose histogram has two peaks*). Otsu binarization can automatically calculates a threshold value from image histogram for a bimodal image by maximizing between-class variance term.

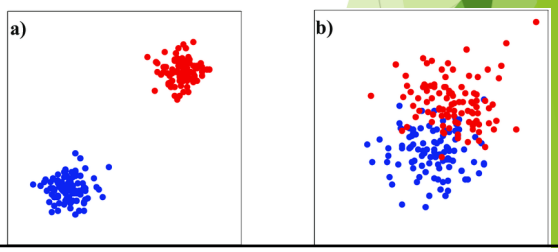


Image Thresholding - Otsu Thresholding

a b
c d

(a) Original image.
(b) Histogram (high peaks were clipped to highlight details in the lower values).
(c) Thresholding results using the basic global algorithm
(d) Result using Otsu's method.

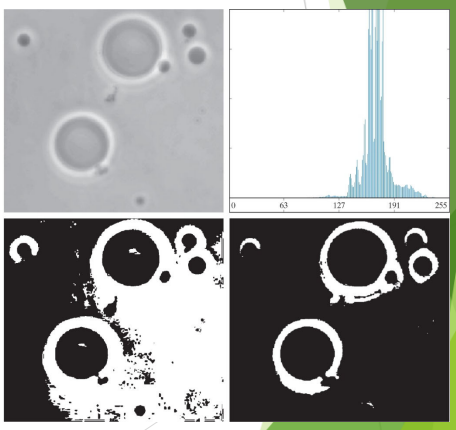
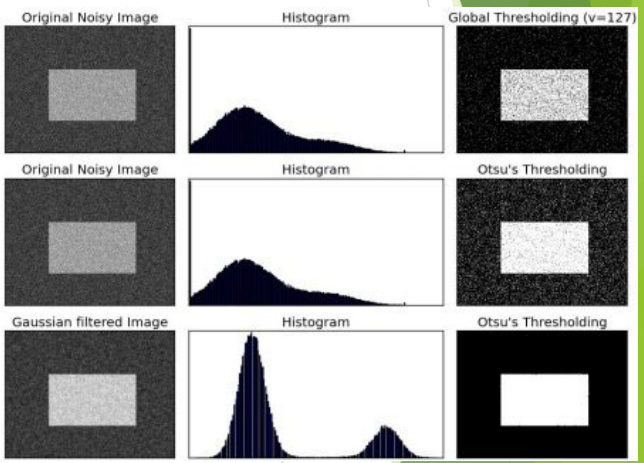


Image Thresholding - Otsu Thresholding

- Global thresholding
- Otsu Thresholding
- Otsu thresholding with Gaussian smoothing

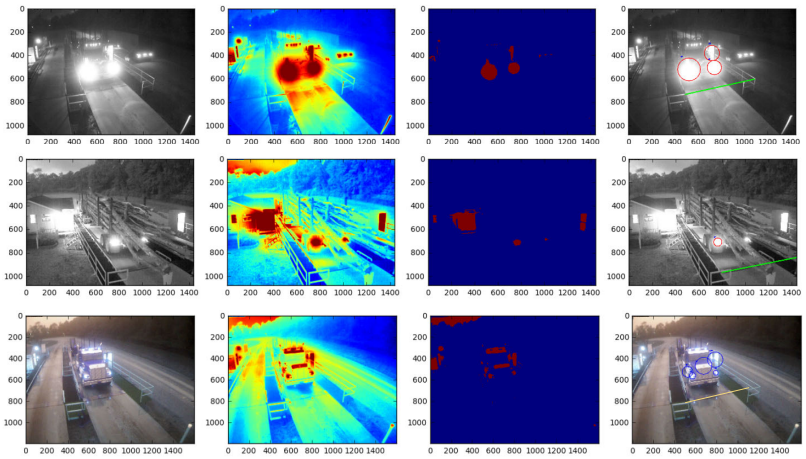


Thresholding Applications

- Detect truck head/tail lights, and compare light's location with scale line
  - Apply different threshold values to detect head/tail lights at night and early morning

Accuracy: 98%

Original → **Blurred** → **Threshold** → **Result**



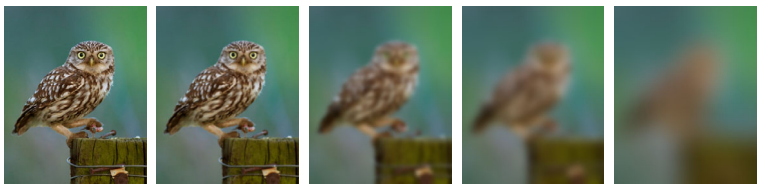
→ Headlight Detection **(Night)**

→ Taillight Detection **(Night)**

→ Headlight Detection **(Early Morning)**

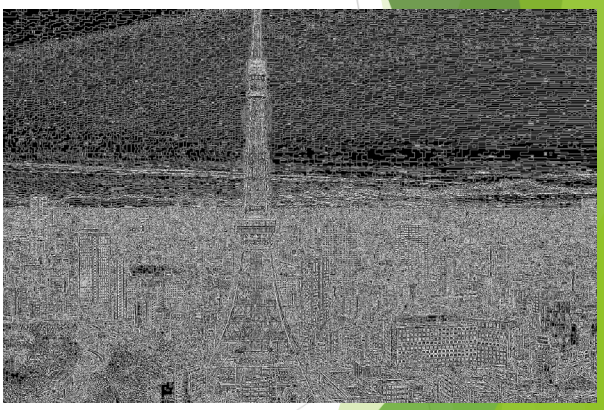
Blurring

- Why blur?
  - Smooths image to remove general noise from an image
  - Can help separate background and foreground features
  - Generally a good step before doing an initial analysis on an image



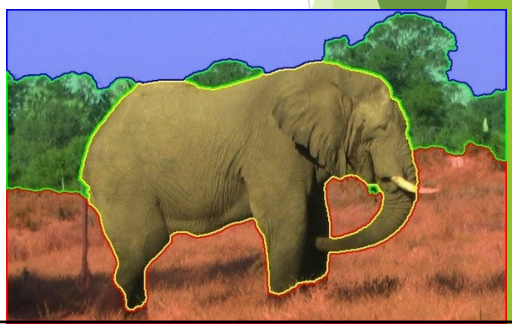
σ = 1 pixel σ = 5 pixels σ = 10 pixels σ = 30 pixels

Spatial – LaPlacian



Feature Extraction

- After an image has been segmented into regions or their boundaries using segmentation methods, the resulting sets of segmented pixels usually have to be converted into a form suitable for further computer processing.
- Typically, the step after segmentation is feature extraction, which consists of feature detection and feature description.



Feature Extraction

- Feature detection refers to finding the features in an image, region, or boundary.
- Feature description assigns quantitative attributes to the detected features.
- For example, we might detect corners in a region boundary, and describe those corners by their orientation and location.
- Question: what is a feature in an image?

Image Features

- In computer vision, image feature refers to two possible entities:
  - A global property of an image or part thereof, for instance the average gray level (global feature); or
  - A part of the image with some special properties, for instance a circle, a line, or a texture region in an intensity image, a planar surface in a range image (local feature)
- Formal definition of Image Features
- Image features are local, meaningful, detectable parts of the image

Purpose of Features

- Features by themselves are seldom generated for human consumption, except in applications such as interactive image processing.
- In fact, some feature extraction methods generate tens, hundreds, or even thousands of descriptor values that would appear meaningless if examined visually.
- Instead, feature description typically is used as a preprocessing step for higher-level tasks, such as image registration, object recognition for automated inspection, searching for patterns (e.g., individual faces and/or fingerprints) in image databases, and autonomous applications, such as robot and vehicle navigation.

Spatial – Background Removal

Spatial – "Foreground" Removal

Spatial – cv2.Laplacian after BG removal

Morphological Transformations

- Leverages kernels/structuring elements to perform simple operations
- Uses any kernel of $M\ X\ N$ shape
- Generally uses odd numbered $M\ X\ N$ shapes where $M = N$
- The values in each kernel should be binary (0 or 1)

- Two basic examples of morphological transformations
  - Erosion
    - "Removes" (chips away) non-zero pixels from non-zero pixel regions, based on the kernel shape
  - Dilating
    - "Expands" non-zero pixels, based on the kernel shape

Morphological Transformations - Erosion

1. Establish a kernel and an "anchor" point (kernel center)
2. Slide kernel across the image from top-left to bottom-right, centering the anchor point on each pixel
3. At each pixel, evaluate the kernel overlaying the image
   - If at each non-zero kernel value, there is a non-zero pixel value on the overlaid image pixel, do nothing
   - Else if **any** non-zero kernel value does not have a corresponding non-pixel value on the overlaid image pixel, set the anchor point pixel of the image to 0

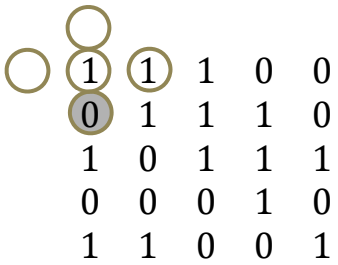
- Final result:
  - Operation will only leave larger clusters of pixels and tend to remove smaller objects that could be associated with noise

Morphological Transformations - Erosion

3x3 Kernel with a cross pattern

0	1	0
1	1	1
0	1	0

Erosion Operation



Final Output

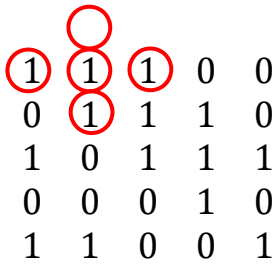
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	0
0	0	0	0	0

Morphological Transformations - Erosion

3x3 Kernel with a cross pattern

0	1	0
1	1	1
0	1	0

Erosion Operation



Final Output

0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	0
0	0	0	0	0

Morphological Transformations - Erosion

3x3 Kernel with a cross pattern

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Erosion Operation

$$\begin{matrix} 1 & 1 & \textcircled{1} & 0 & 0 \\ 0 & \textcircled{1} & \textcircled{1} & \textcircled{1} & 0 \\ 1 & 0 & \textcircled{1} & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{matrix}$$

Final Output

$$\begin{matrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \textcircled{1} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Morphological Transformations - Erosion

3x3 Kernel with a cross pattern

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Erosion Operation

$$\begin{matrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & \textcircled{1} & 0 \\ 1 & 0 & \textcircled{1} & \textcircled{1} & \textcircled{1} \\ 0 & 0 & 0 & \textcircled{1} & 0 \\ 1 & 1 & 0 & 0 & 1 \end{matrix}$$

Final Output

$$\begin{matrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \textcircled{1} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Morphological Transformations - Erosion

3x3 Kernel with a cross pattern

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Erosion Operation

$$\begin{matrix} 1 & \textcircled{1} & 1 & 0 & 0 \\ 0 & 1 & \textcircled{1} & 1 & 0 \\ 1 & 0 & 1 & \textcircled{1} & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{matrix}$$

Final Output

$$\begin{matrix} 0 & \textcircled{1} & 0 & 0 & 0 \\ 0 & 0 & \textcircled{1} & 0 & 0 \\ 0 & 0 & 0 & \textcircled{1} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Morphological Transformations - Dilation

1. Establish a kernel and an “anchor” point (kernel center)
2. Slide kernel across the image from top-left to bottom-right, centering the anchor point on each pixel
3. At each pixel, evaluate the kernel overlaying the image
   - At each non-zero kernel value, set the corresponding overlaid image pixels to 1

- Final result:
  - Operation will expand non-zero pixels into their surrounding regions, resulting in an image with larger clusters of non-zero pixels

Morphological Transformations - Dilation

3x3 Kernel with a cross pattern

0	1	0
1	1	1
0	1	0

Dilation Operation

0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	0
0	0	0	0	0

Final Output

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	0	1	0
0	0	0	0	0

Morphological Transformations - Dilation

3x3 Kernel with a cross pattern

0	1	0
1	1	1
0	1	0

Dilation Operation

0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	0
0	0	0	0	0

Final Output

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	0	1	0
0	0	0	0	0

Morphological Transformations - Dilation

3x3 Kernel with a cross pattern

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Dilation Operation

$$\begin{matrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \textcircled{1} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Final Output

$$\begin{matrix} 1 & 1 & \textcircled{1} & 0 & 0 \\ 0 & \textcircled{1} & \textcircled{1} & \textcircled{1} & 0 \\ 0 & 0 & \textcircled{1} & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Morphological Transformations - Dilation

3x3 Kernel with a cross pattern

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Erosion Operation

$$\begin{matrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \textcircled{1} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Final Output

$$\begin{matrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & \textcircled{1} & 0 \\ 0 & 0 & \textcircled{1} & \textcircled{1} & \textcircled{1} \\ 0 & 0 & 0 & \textcircled{1} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Morphological Transformations - Dilation

3x3 Kernel with a cross pattern

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{matrix}$$

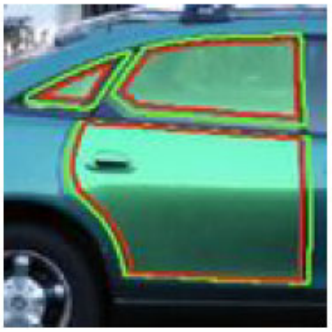
Dilation Operation

$$\begin{matrix} 0 & \textcircled{1} & 0 & 0 & 0 \\ 0 & 0 & \textcircled{1} & 0 & 0 \\ 0 & 0 & 0 & \textcircled{1} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

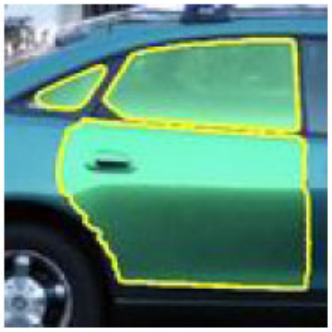
Final Output

$$\begin{matrix} 1 & \textcircled{1} & 1 & 0 & 0 \\ 0 & 1 & \textcircled{1} & 1 & 0 \\ 0 & 0 & 1 & \textcircled{1} & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Dilation



(a) Initialization.



(b) Result.

Kernels

Image Erosion

Pattern Matching
Kernel (25x25):

Array of 25 x 25 ones

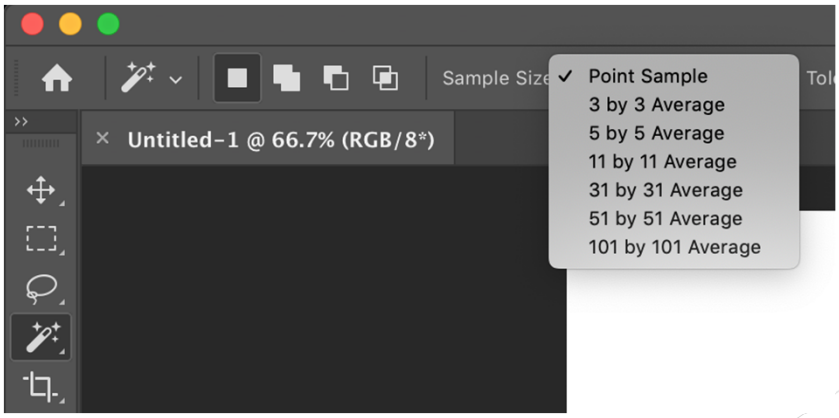
Kernels

Image Erosion

Pattern Matching
Kernel (25x25):

Array of 25 x 25 ones

Show Photoshop



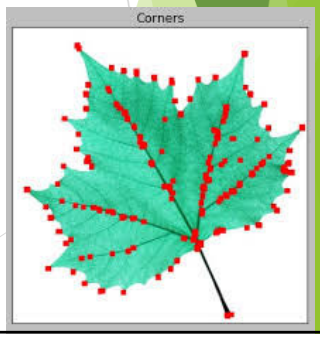
Masks

- Binary black and white images
- Bitwise operations, such as bitwise_and can be used to obtain parts of the original image
- Most segmentation operations use masks instead of the original image

Harris Corner Detection

- Corners are regions in the image with large variation in intensity in all the directions.
- One early attempt to find these corners was done by Chris Harris & Mike Stephens in their paper A Combined Corner and Edge Detector in 1988, so now it is called Harris Corner Detector.
- The approach is to maximize

$$C(x,y) = \sum_s \sum_t w(s,t)[f(s+x,t+y) - f(s,t)]^2$$



Harris Corner Detection

$$C(x,y) = \sum_s \sum_t w(s,t)[f(s+x,t+y) - f(s,t)]^2$$

- The weighting function $w(x,y)$ used in the HS detector generally has one of two forms:
  - It is 1 inside the patch and 0 elsewhere (i.e., it has the shape of a box lowpass filter kernel), or
  - It is an exponential function of the form

$$w(s,t) = e^{-(s^2+t^2)/2\sigma^2}$$

- The box is used when computational speed is paramount and the noise level is low. The exponential form is used when data smoothing is important.

Harris Corner Detection

$$C(x,y) = \sum_s \sum_t w(s,t)[f(s+x,t+y) - f(s,t)]^2$$

$$C(x,y) = \begin{bmatrix} x & y \end{bmatrix} M \begin{bmatrix} x \\ y \end{bmatrix}$$

Where,

$$M = \sum_s \sum_t w(s,t) \begin{bmatrix} f_x^2 & f_x f_y \\ f_x f_y & f_y^2 \end{bmatrix}$$

Harris Corner Detection

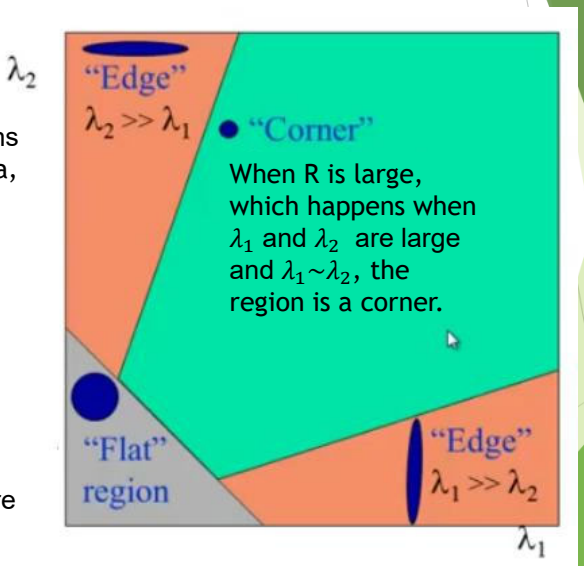
- $R = \det(M) - k\big(trace(M)\big)2$
  - $\det(M) = \lambda_1 \lambda_2$
  - $\text{trace}(M) = \lambda_1 \lambda_2$
- λ_1 and λ_2 are eigenvalues of M.
- Constant k is determined empirically, and its range of values depends on the implementation; In general, the smaller it is, the more likely to find corners.

Harris Corner Detector

$$R = \det(M) - k\big(trace(M)\big)^2$$

When $R < 0$, which happens when $\lambda_1 \gg \lambda_2$ or vice versa, the region is edge.

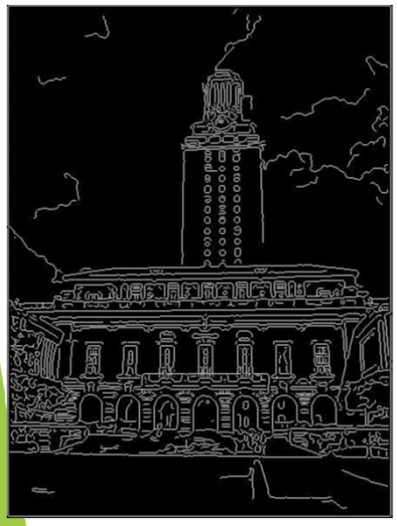
When $|R|$ is small, which happens when λ_1 and λ_2 are small, the region is flat.



Hough Transform

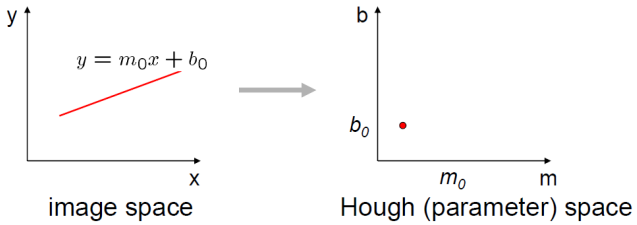
- A popular algorithm to detect lines and simple curves.

Hough Transform



- Hough Transform is a voting technique that answers these questions:
  - Given points that belong to a line, what is the line?
  - How many lines are there?
  - Which points belong to which lines?
- General Methodology:
  - Record vote for each possible line on which each edge point lies.
  - Look for lines that get many votes.

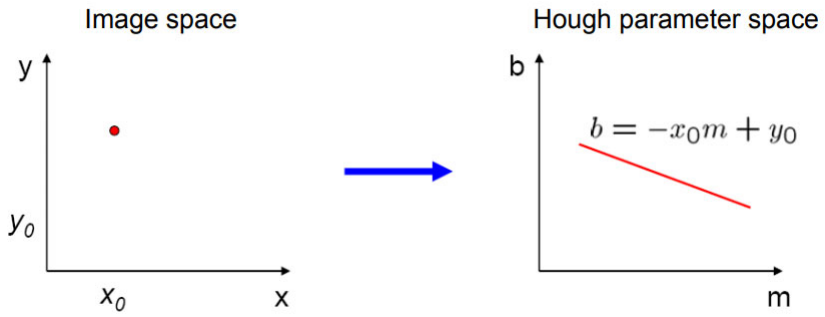
Hough Transform



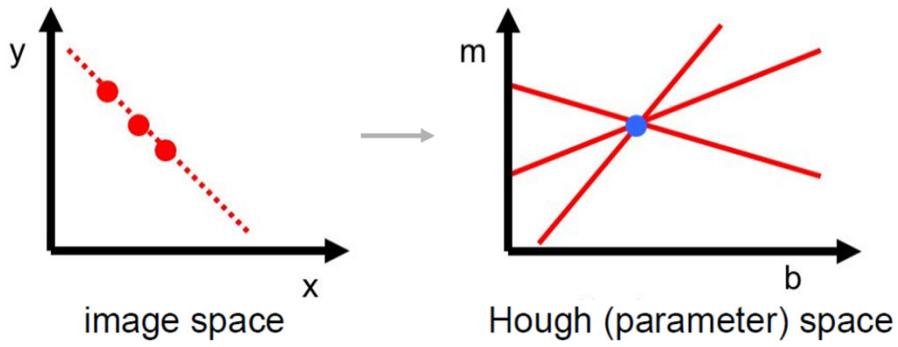
Connection between image (x,y) and Hough (m,b) spaces

- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points (x,y), find all (m,b) such that y = mx + b

Hough Transform



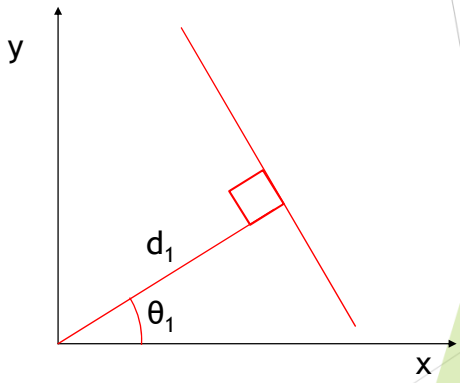
Hough Transform



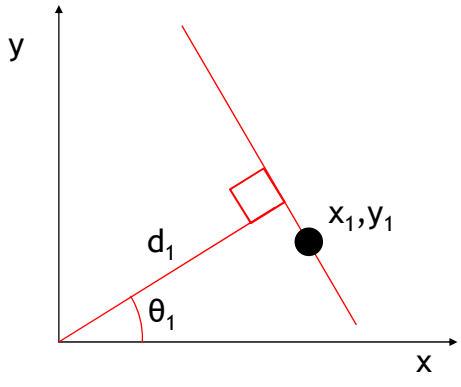
P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures*, Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

The Hough Transform

- Use polar representation to avoid infinite slope

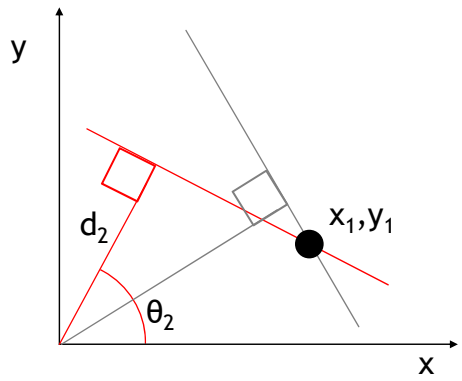


The Hough Transform



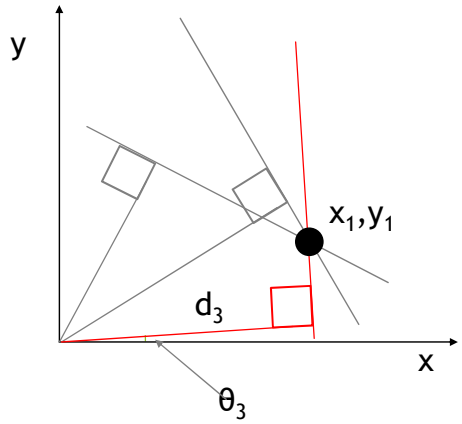
$$d = x_1 cos\theta + y_1 sin\theta$$

The Hough Transform



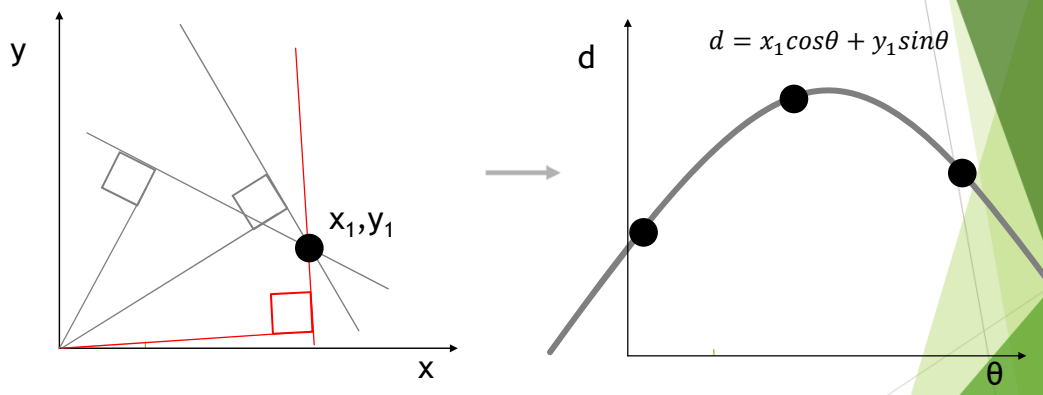
$$d = x_1 cos\theta + y_1 sin\theta$$

The Hough Transform



$$d = x_1 cos\theta + y_1 sin\theta$$

The Hough Transform



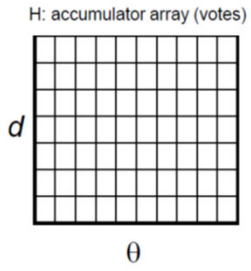
Hough Transform

Using the polar parameterization:

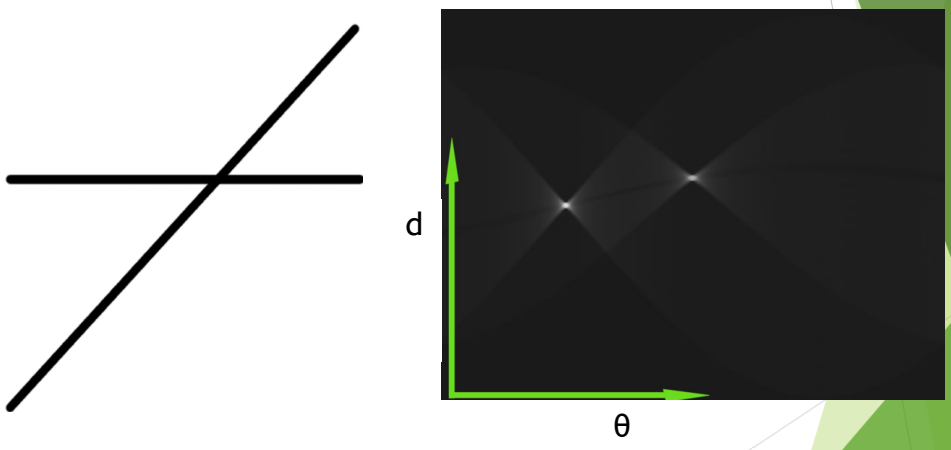
$$x\cos\theta - y\sin\theta = d$$

Basic Hough transform algorithm

1. Initialize H[d, θ]=0
2. for each edge point I[x,y] in the image
   for θ = [θ_{min} to θ_{max}] // some quantization
   $d = x\cos\theta - y\sin\theta$
   H[d, θ] += 1
3. Find the value(s) of (d, θ) where H[d, θ] is maximum
4. The detected line in the image is given by $d = x\cos\theta - y\sin\theta$



The Hough Transform (HT)

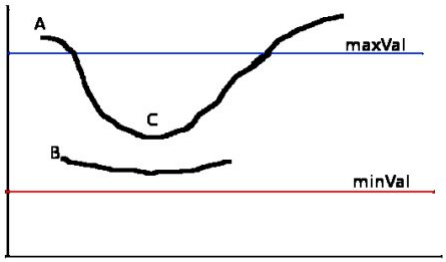


Contours and Edge Detection - findContours

- Contours are a set of edges that are enclosed
- findContours produces coordinate values for each set of contours
- Useful in obtaining features and manipulating results obtained from other edge detection algorithms

Contours and Edge Detection - Canny

- Uses the sobel x, sobel y algorithms to obtain edges in both directions
- Determines the center of each edge and transforms the edges into a single pixel wide line
- Uses hysteresis thresholding, on the gradient intensity, to obtain the final valid edges
  - Any line below the minVal is dropped
  - Any line above maxVal is kept
  - Any line between minVal and maxVal is separately evaluated



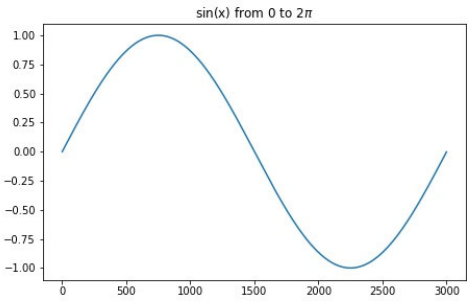
Contours and Edge Detection - boundingRect

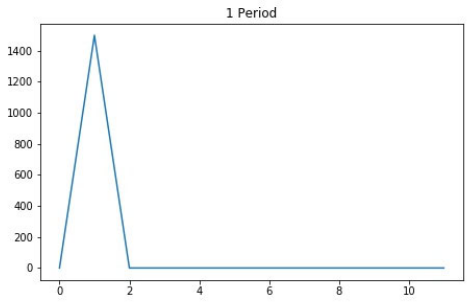
- Finds the straight edge, unrotated, rectangle that bounds the contours
- Obtains:
  - x, y coordinates of the top left corner of the box
  - height, width of the rectangle
- Important to note:
  - OpenCV lists the coordinates as x, y
  - NumPy lists the coordinates as y, x (row, column)

Cropping

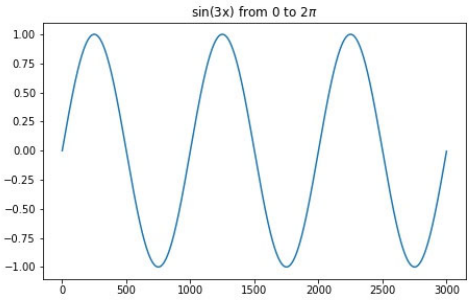
- Cropping can be achieved through NumPy.
- Image[y_i:y_f, x_i:x_f]
  - For channel specific: Image[y_i:y_f, x_i:x_f, channel]
- Things to remember:
  - NumPy uses y, x format
  - When subsetting, NumPy does [i, f), so i is inclusive, but f is exclusive

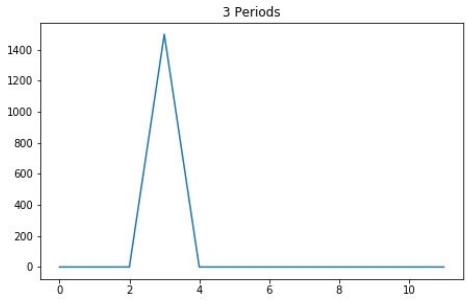
Frequency Domain - Fourier Transform



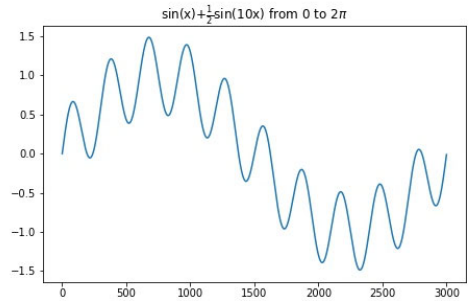


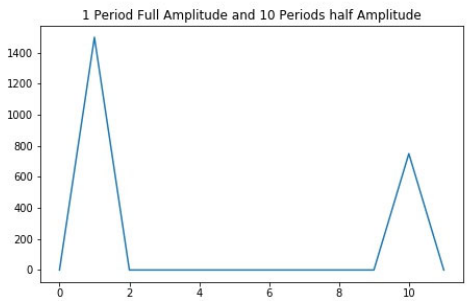
Frequency Domain - Fourier Transform



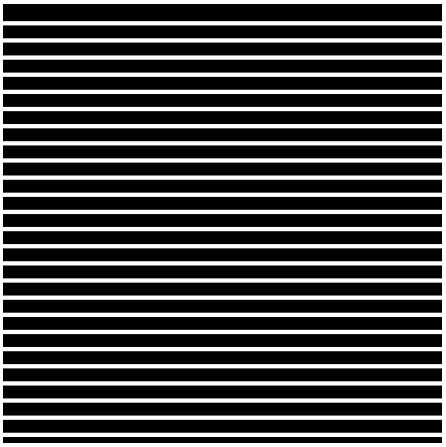


Frequency Domain - Fourier Transform

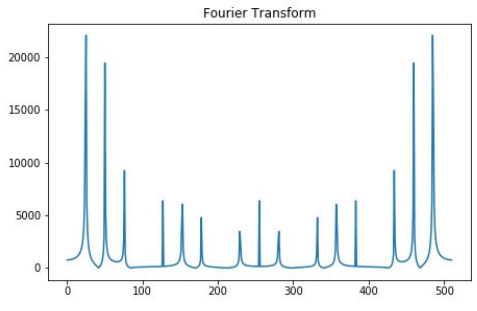


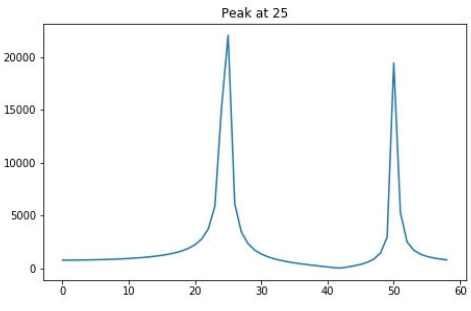


Frequency Domain
- Fourier Transform



Frequency Domain
- Fourier Transform

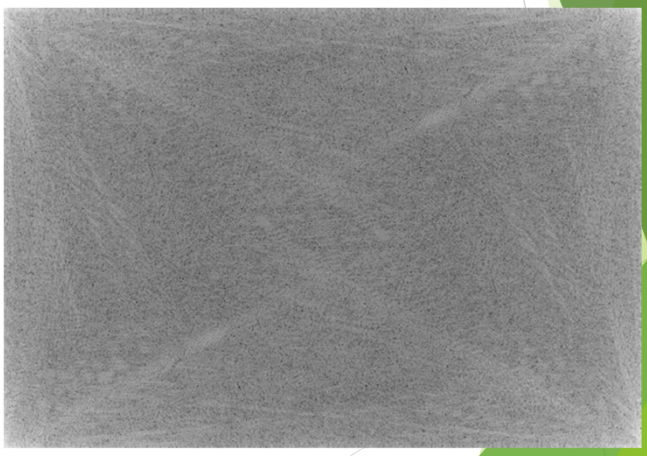




Fourier Transforms

FFT in Two Dimensions

- Output plots frequency_x, frequency_y, power (amplitude)



Fourier Transforms

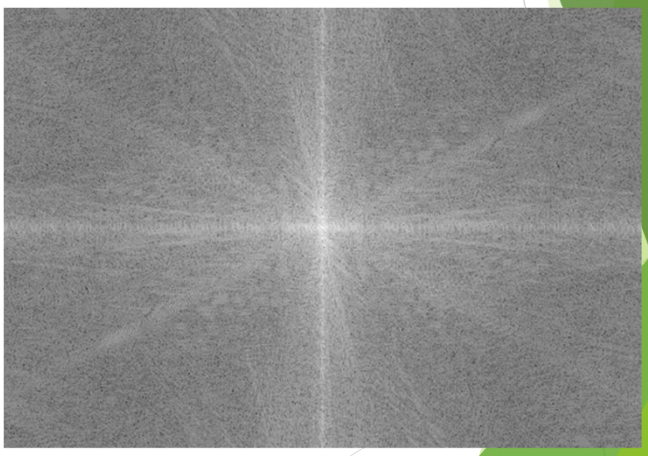
FFT in Two Dimensions

- Shifting centers the low frequencies to the center
- FFTs can be altered to generate high-pass and low-pass filters
- Manipulations in the Fourier domain can be approximately accomplished in the spatial domain and vice versa

Fourier Transforms

FFT in Two Dimensions

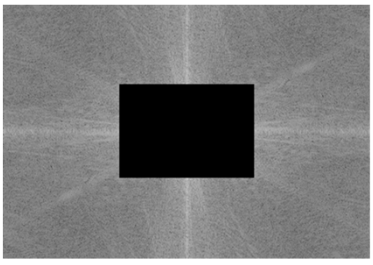
- Shifted FFT image



Fourier Transforms

FFT in Two Dimensions

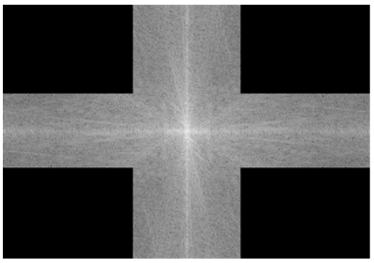
- High Pass Filter



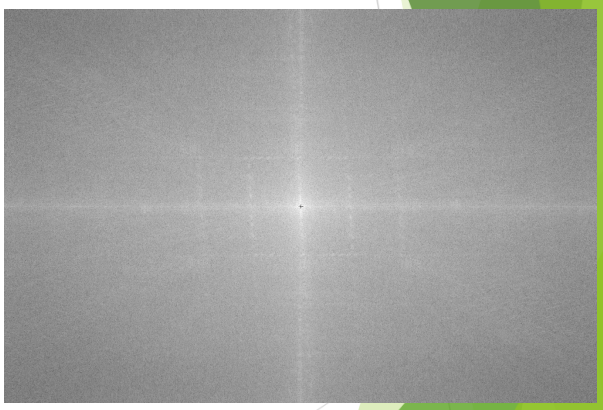
Fourier Transforms

FFT in Two Dimensions

- Low Pass Filter

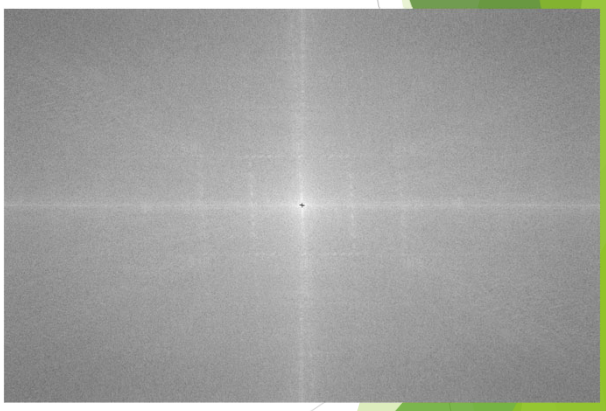


Spatial Vs Frequency



https://www.burgessyachts.com/en/locations/tokyo/

Frequency



Spatial – Kernel