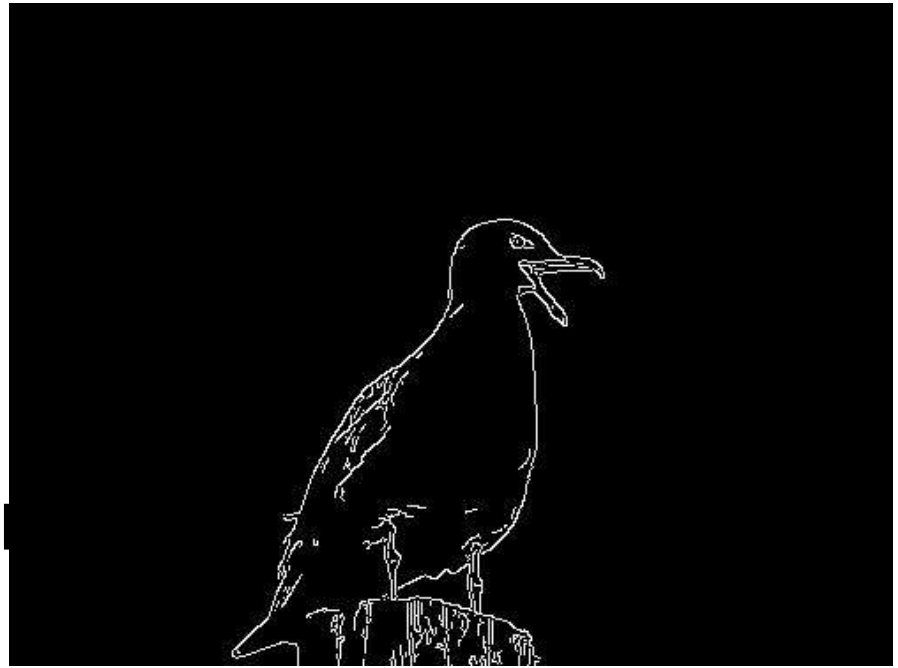
Computer Vision for Embedded Systems

Yung-Hsiang Lu
Purdue University
yunglu@purdue.edu

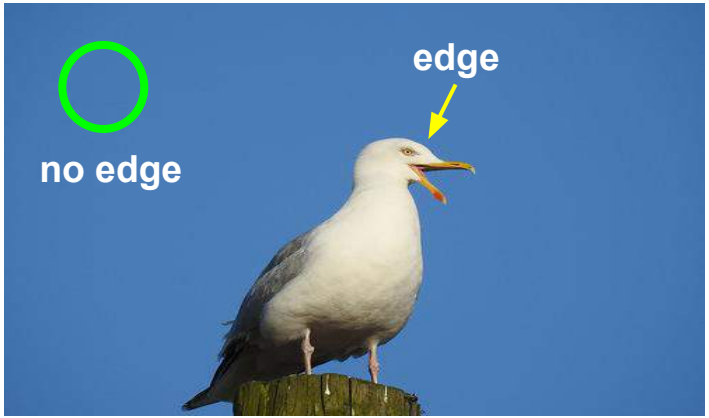
PURDUE UNIVERSITY

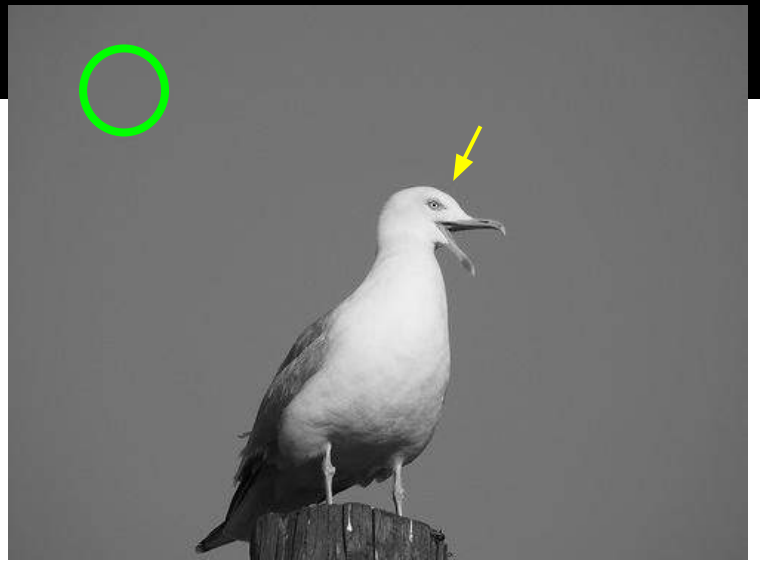
OpenCV

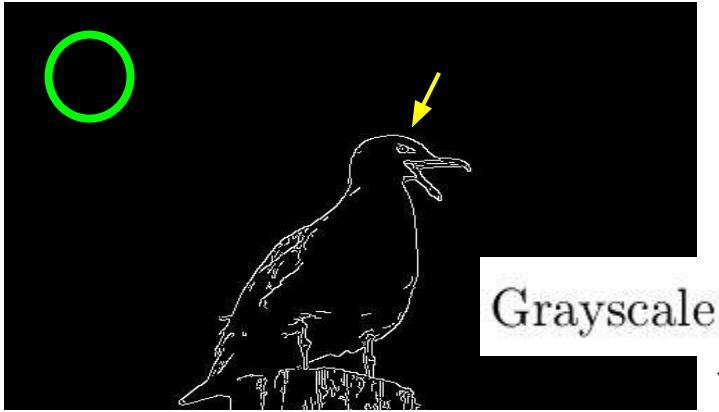
Edge Detection



What is an edge ?





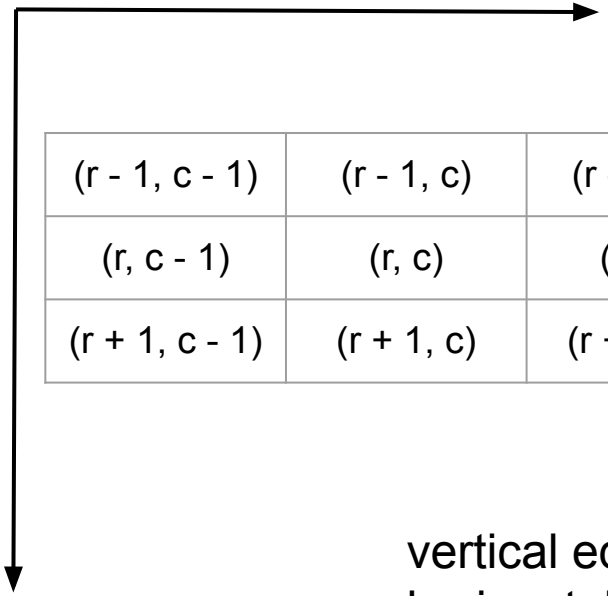


$$\text{Grayscale} = \frac{1}{3}(\text{Red} + \text{Green} + \text{Blue})$$ ✗

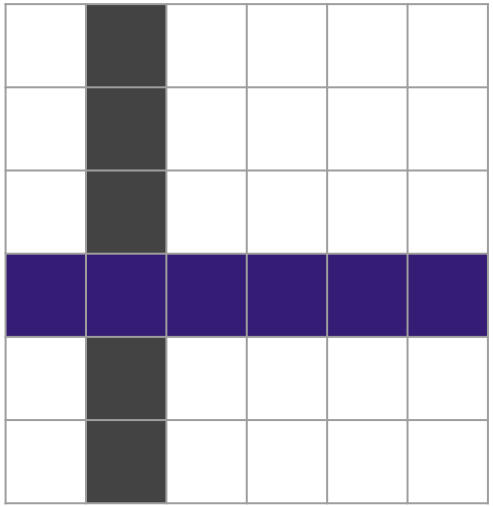
adjust for brightness of different colors

$$\text{Grayscale} = 0.30 \times \text{Red} + 0.59 \times \text{Green} + 0.11 \times \text{Blue}$$

Define Edge



(r - 1, c - 1)	(r - 1, c)	(r - 1, c + 1)
(r, c - 1)	(r, c)	(r, c + 1)
(r + 1, c - 1)	(r + 1, c)	(r + 1, c + 1)



vertical edge if $|i(r,c) - i(r,c - 1)| > t$

horizontal edge if $|i(r,c) - i(r - 1,c)| > t$

i is the intensity

t is the threshold

$|i(r,c) - i(r,c-1)| > t$

$|i(r,c) - i(r-1,c)| > t$

define $\Delta_r = 1$ and $\Delta_c = 1$

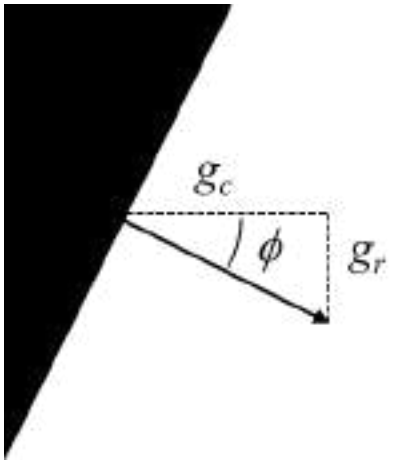
$$|\frac{i(r,c) - i(r-\Delta_r,c)}{\Delta_r}| > t \qquad |\frac{i(r,c) - i(r,c-\Delta_c)}{\Delta_c}| > t$$

$$|\frac{\partial i}{\partial r}| > t \qquad |\frac{\partial i}{\partial c}| > t$$

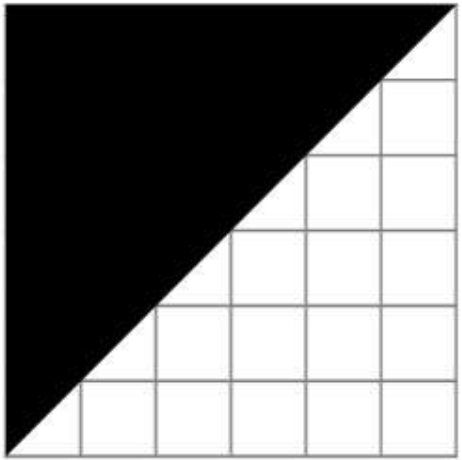
$$\frac{\partial i}{\partial r}\vec{r} + \frac{\partial i}{\partial c}\vec{c} = \nabla i$$

$$\frac{\partial i}{\partial r}\vec{r} + \frac{\partial i}{\partial c}\vec{c} = \nabla i = g_r\vec{r} + g_c\vec{c}$$

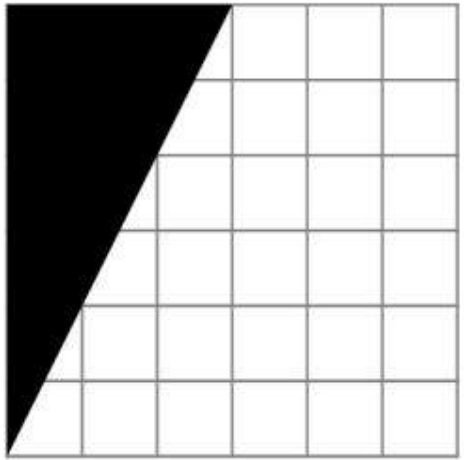
$$|\nabla i| = |g_r\vec{r} + g_c\vec{c}| = \sqrt{g_r^2 + g_c^2} > t$$



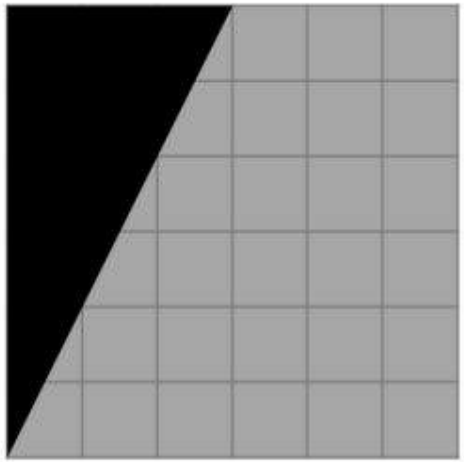
$$\tan(\phi) = \frac{g_r}{g_c}$$



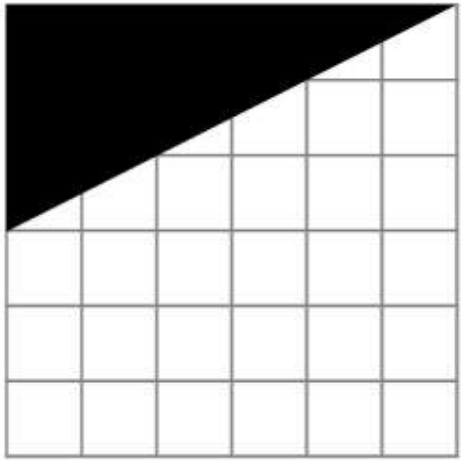
(a) $\alpha\vec{r} + \alpha\vec{c}$



(b) $\alpha\vec{r} + 2\alpha\vec{c}$



(c) $\beta\vec{r} + 2\beta\vec{c}$

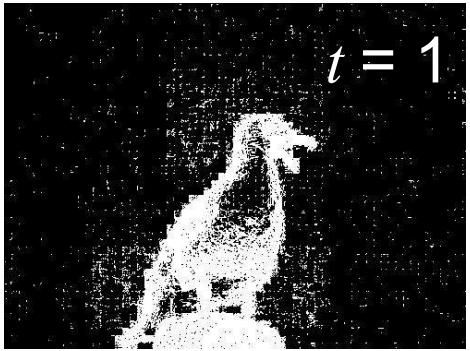


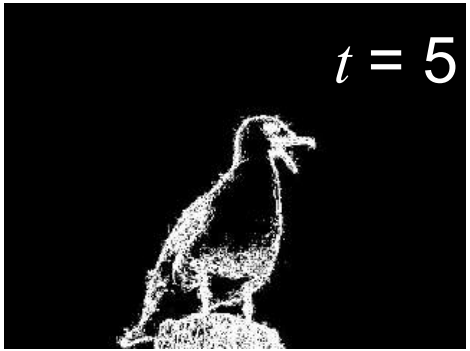
(d) $2\alpha\vec{r} + \alpha\vec{c}$

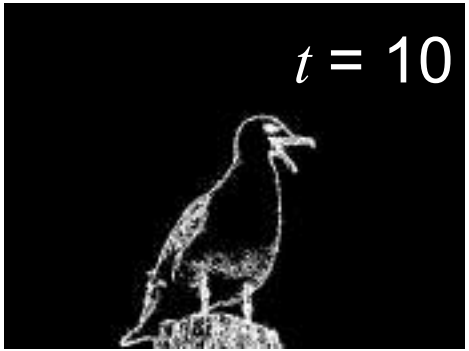
Edge Thresholds & Canny Detector

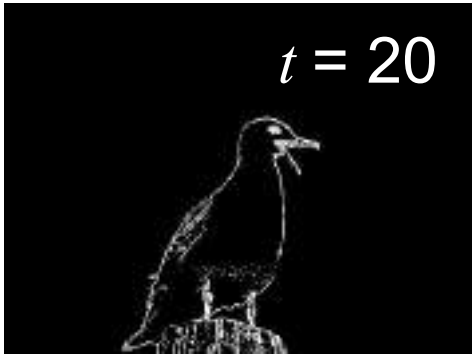
```
# edge01.py
# detect edges in an image
# conver RGB image to gray
# detect sudden changes of intensity as edges
# save the gray image and the edge image

import numpy
import sys
import cv2

def detectedge(filename, threshold):
    namebase = filename.replace('.JPEG', '')
    rgbimage = cv2.imread(filename)
    grayimage = cv2.cvtColor(rgbimage, cv2.COLOR_BGR2GRAY)
    grayname = namebase + '_gray.jpg'
    cv2.imwrite(grayname, grayimage)
    # print (grayimage.shape)
    [height, width] = grayimage.shape
    edgeimage = numpy.zeros([height, width])
    for row in range(1, height):
        for col in range(1, width):
            # print ([row, col])
            rowdiff = int(grayimage[row, col]) - int(grayimage[row - 1, col])
            rowdiff = abs(rowdiff)
            coldiff = int(grayimage[row, col]) - int(grayimage[row, col - 1])
            coldiff = abs(coldiff)
            if ((rowdiff > threshold) or (coldiff > threshold)):
                edgeimage[row, col] = 255
    edgename = namebase + '_edge' + str(threshold) + '.jpg'
    cv2.imwrite(edgename, edgeimage)

if __name__ == "__main__":
    if (len(sys.argv) != 2):
        sys.exit('need the name of an image file')
    thresholds = [1, 5, 10, 20, 40, 80, 160]
    for th in thresholds:
        detectedge(sys.argv[1], th)
```

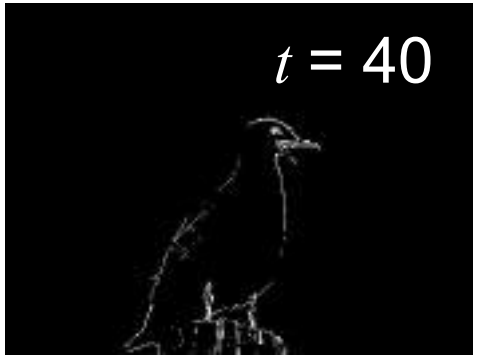
Does this work?





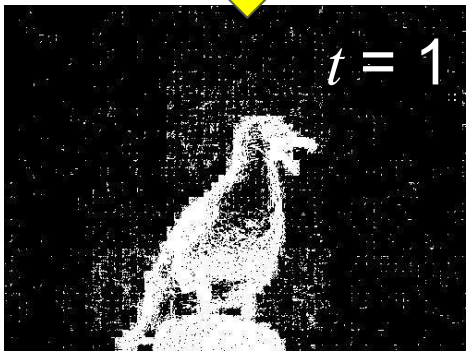


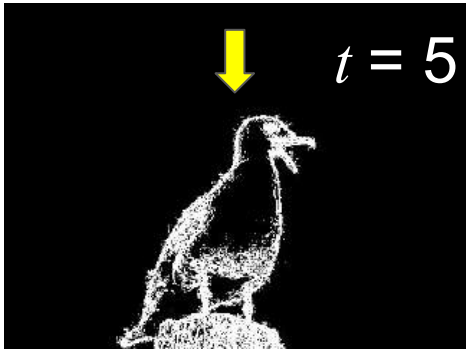


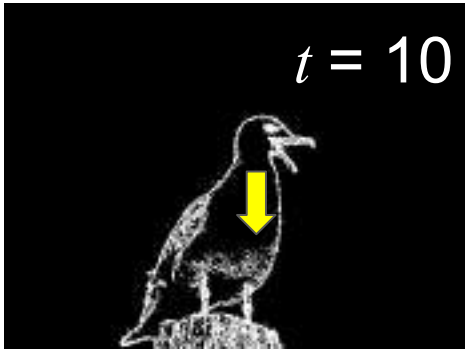


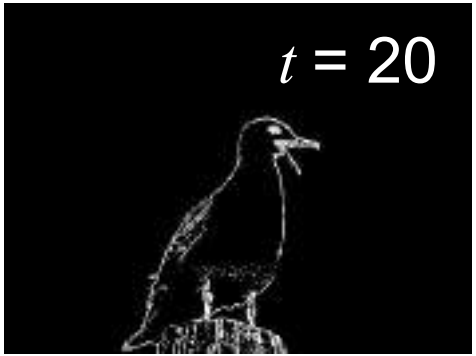
Does this work?

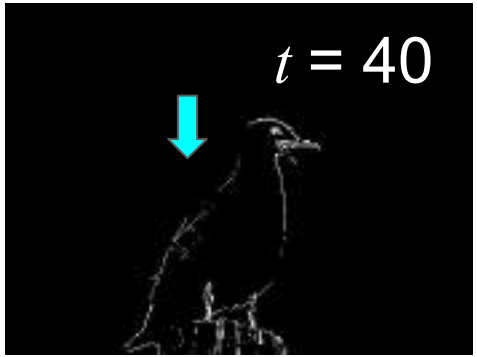
noise



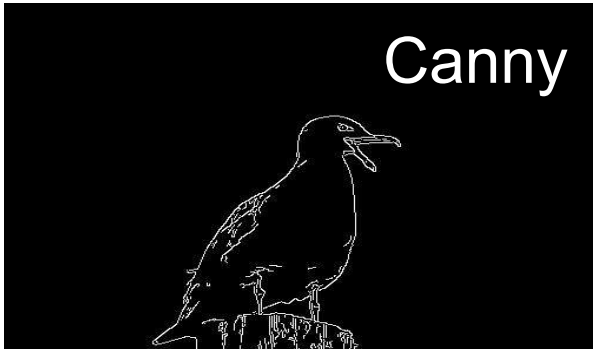




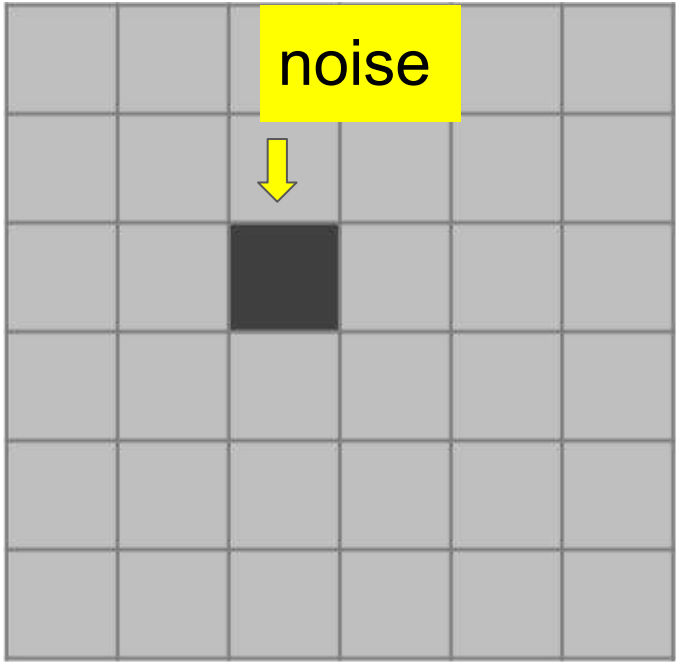




t too high

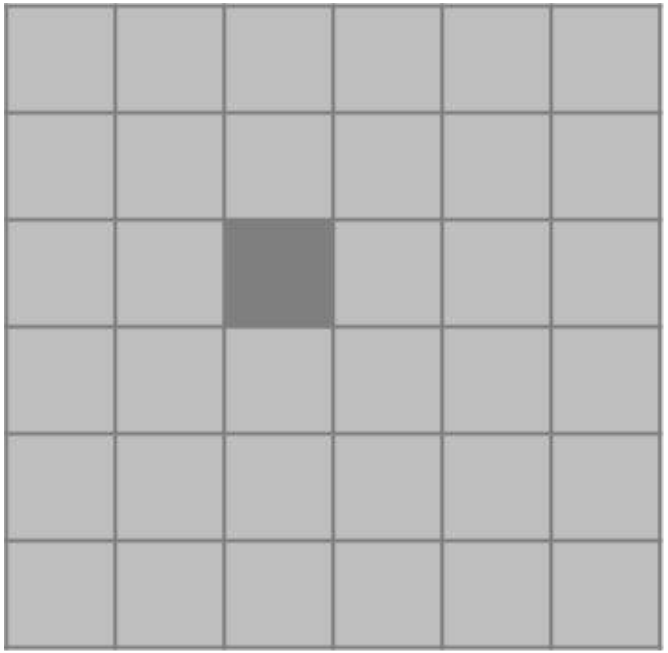


Noise and Smoothening



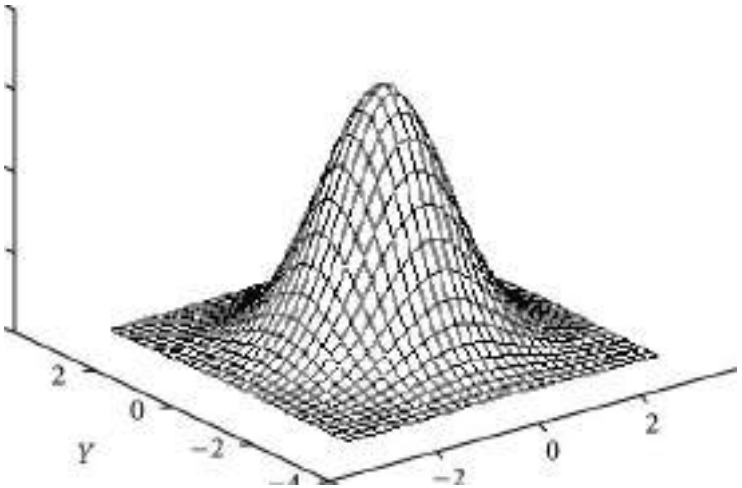
smoothening

$$U_3 = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



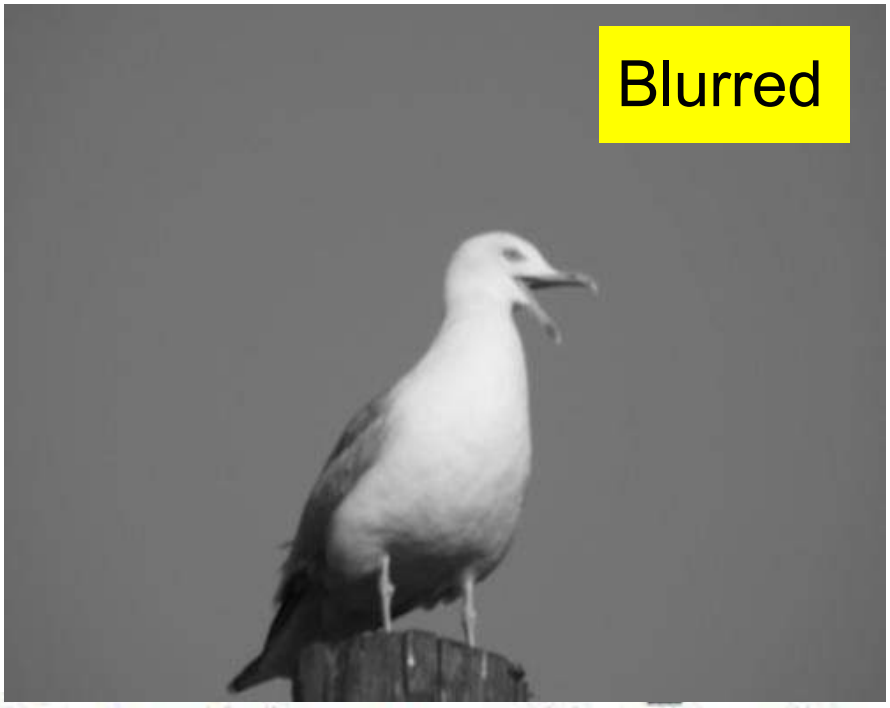
3 x 3 uniform filter

2D Gaussian Filter



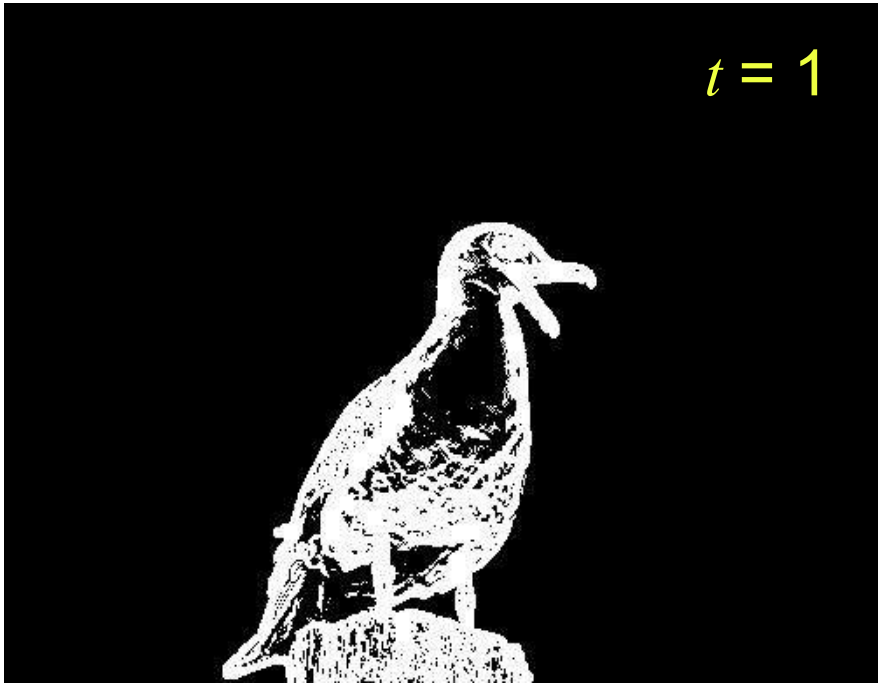
$$G = \frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

$$G = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 26 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$




```
blurimage = cv2.GaussianBlur(grayimage, (5, 5), 0)
```

Gray Level



Blurred

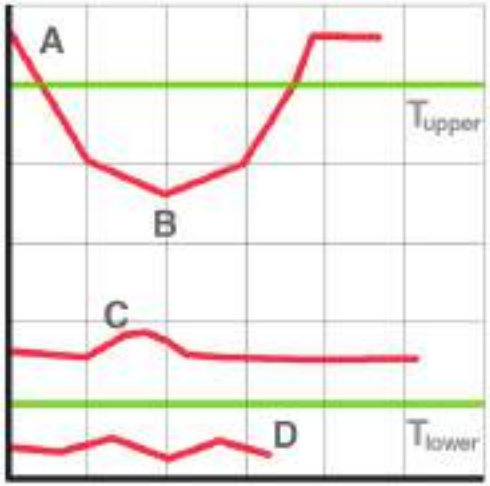
Gray Level

Blurred

Canny Edge Detector

1. 5 x 5 Gaussian filter to reduce noise
2. Sobel filters

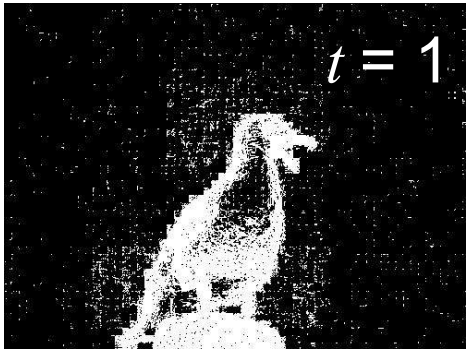
$$G_r = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} G_c = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

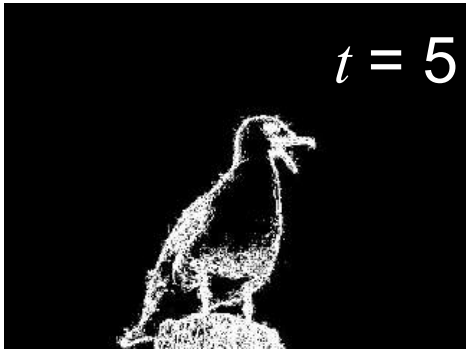


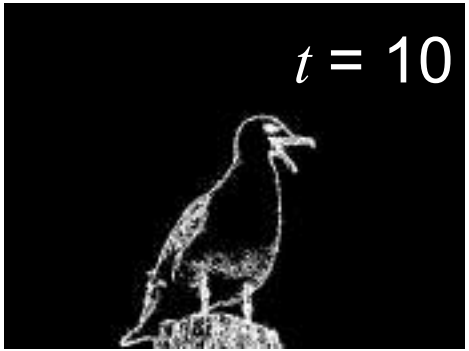
3. adaptive thresholds
   a. A is an edge pixel
   b. B is an edge pixel because of A
   c. C, D are not edge pixels

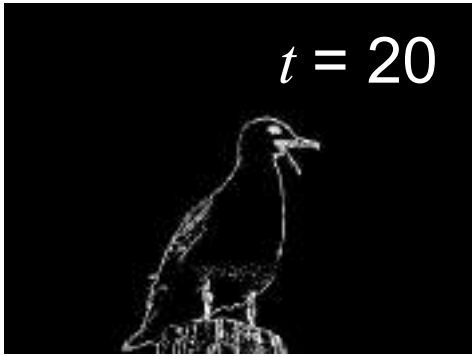
https://www.pyimagesearch.com/2021/05/12/opencv-edge-detection-cv2-canny/

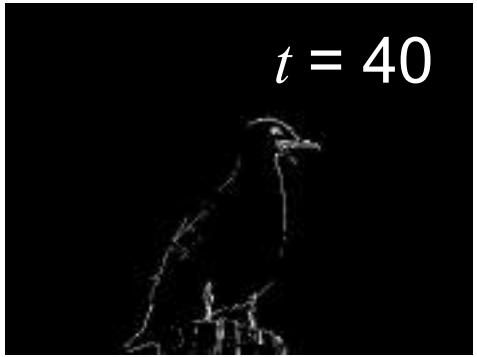
Does this work?







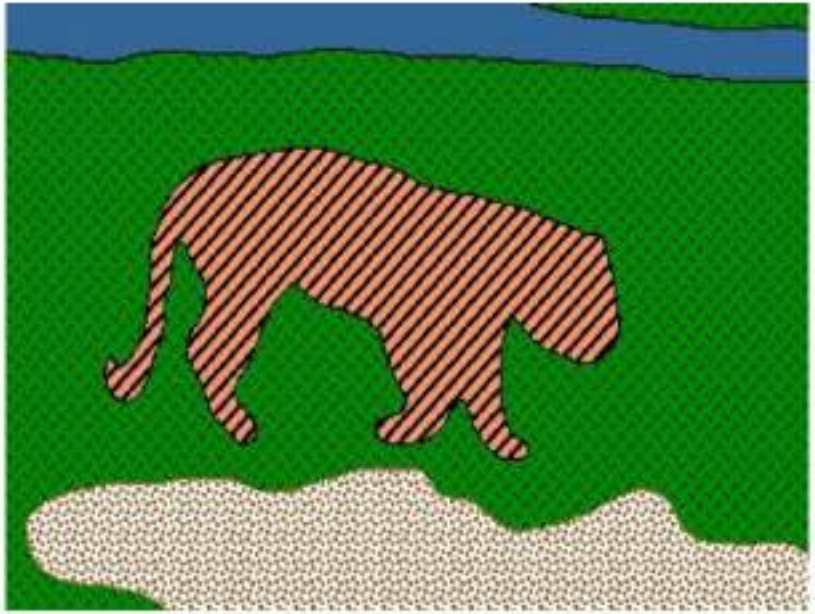




Canny Edge Detector

```
import numpy
import sys
import cv2

def detectedge(filename):
    namebase = filename.replace('.JPEG', '')
    image = cv2.imread(filename)
    edges = cv2.Canny(image, 100, 200)
    edgename = namebase + '_canny.jpg'
    cv2.imwrite(edgename, edges)

if __name__ == "__main__":
    if (len(sys.argv) != 2):
        sys.exit('need the name of an image file')
    detectedge(sys.argv[1])
```

Segmentation



http://vision.stanford.edu/teaching/cs131_fall1617

https://github.com/nightrome/cocostuff

Virtual background in video call

https://www.lifesize.com/en/blog/virtual-backgrounds/

Mean Shift Clustering

```
shifted = cv2.pyrMeanShiftFiltering(image, 21, 51)
```

Original

Segmentation

Performs initial step of meanshift segmentation of an image.

C++: void **pyrMeanShiftFiltering**(InputArray **src**, OutputArray **dst**, double **sp**, double **sr**, int **maxLevel**=1, TermCriteria **termcrit**=TermCriteria(TermCriteria::MAX_ITER+TermCriteria::EPS,5,1))

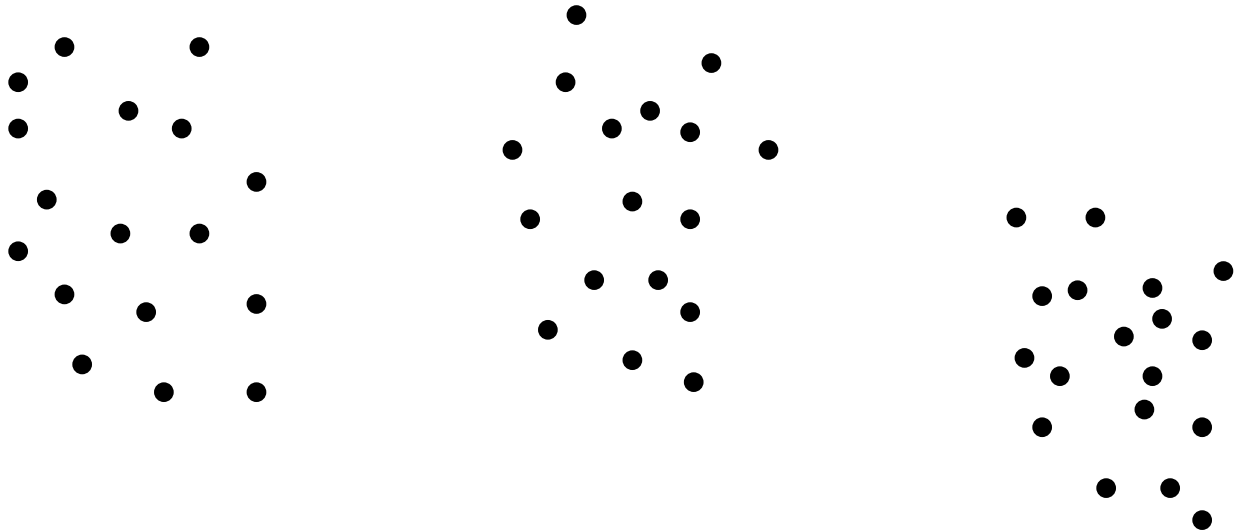
Python: cv2.**pyrMeanShiftFiltering**(src, sp, sr[, dst[, maxLevel[, termcrit]]]) → dst

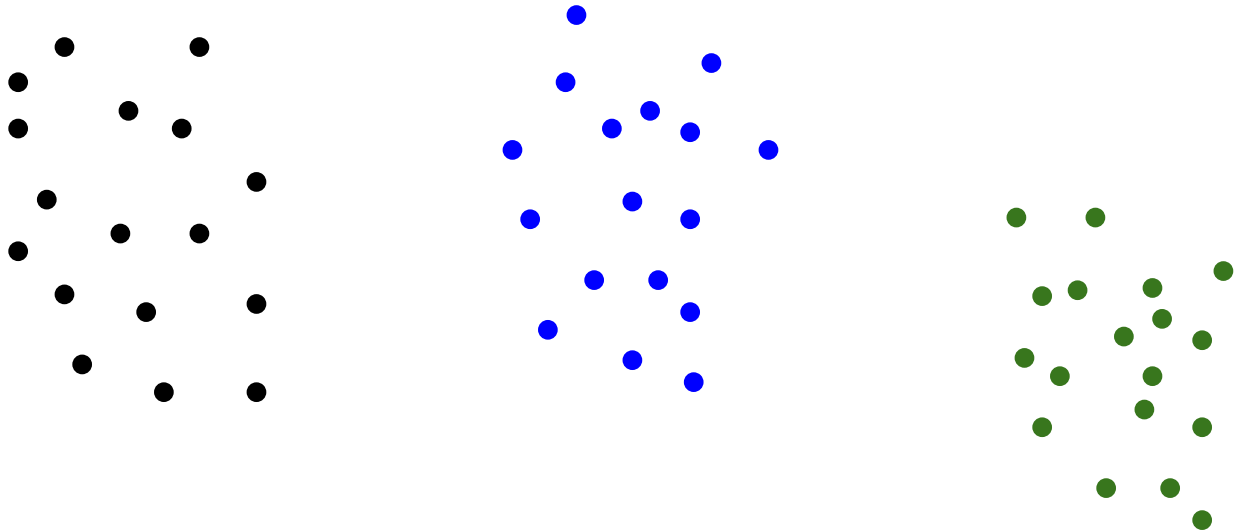
C: void **cvPyrMeanShiftFiltering**(const CvArr* **src**, CvArr* **dst**, double **sp**, double **sr**, int **max_level**=1, CvTermCriteria **termcrit**=cvTermCriteria(CV_TERMCRIT_ITER+CV_TERMCRIT_EPS,5,1))

Python: cv.**PyrMeanShiftFiltering**(src, dst, sp, sr, maxLevel=1, termcrit=(CV_TERMCRIT_ITER+CV_TERMCRIT_EPS, 5, 1)) → None

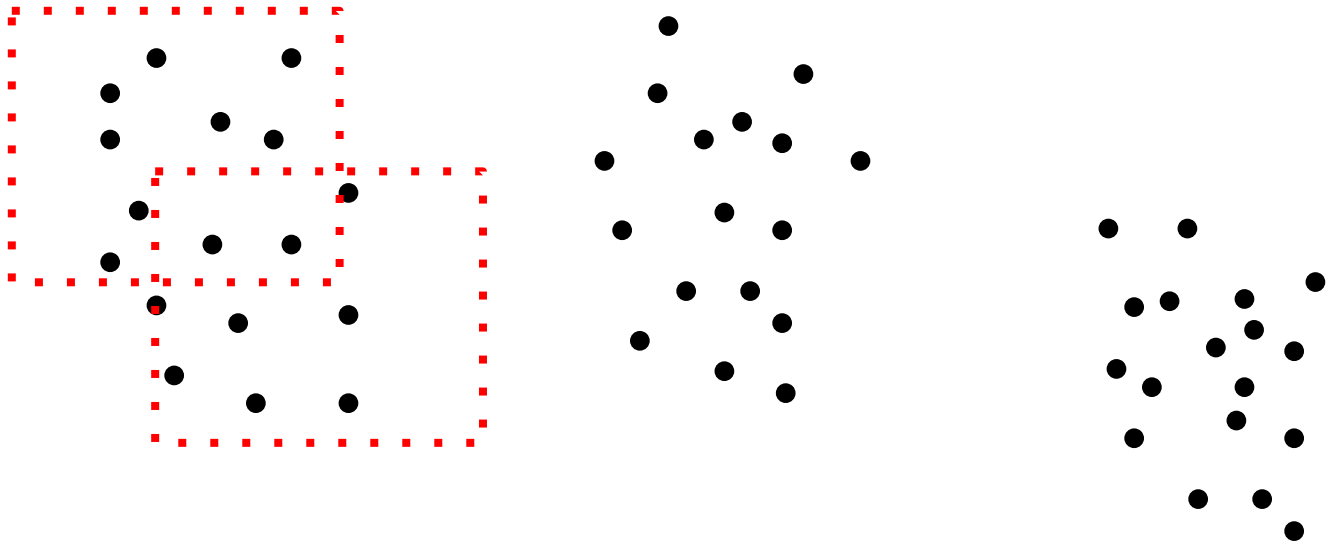
Parameters:
src – The source 8-bit, 3-channel image.
dst – The destination image of the same format and the same size as the source.
sp – The spatial window radius.
sr – The color window radius.
maxLevel – Maximum level of the pyramid for the segmentation.
termcrit – Termination criteria: when to stop meanshift iterations.

http://www.opencv.org.cn/opencvdoc/2.3.2/html/modules/imgproc/doc/filtering.html#pyrmeanshiftfiltering





$$(x, y) : X - sp \leq x \leq X + sp, Y - sp \leq y \leq Y + sp, \|(R, G, B) - (r, g, b)\| \leq sr$$



Erosion and Dilation

Segmentation - Identifying Silkscreen on PCB

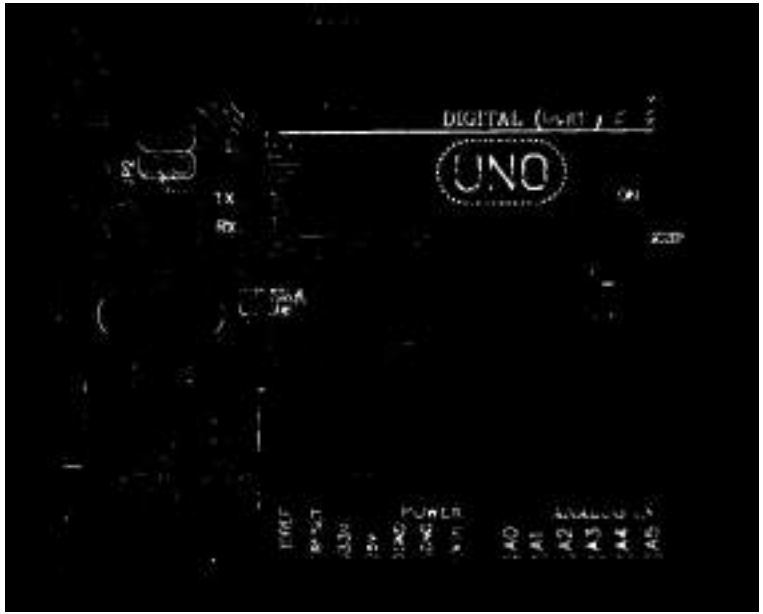
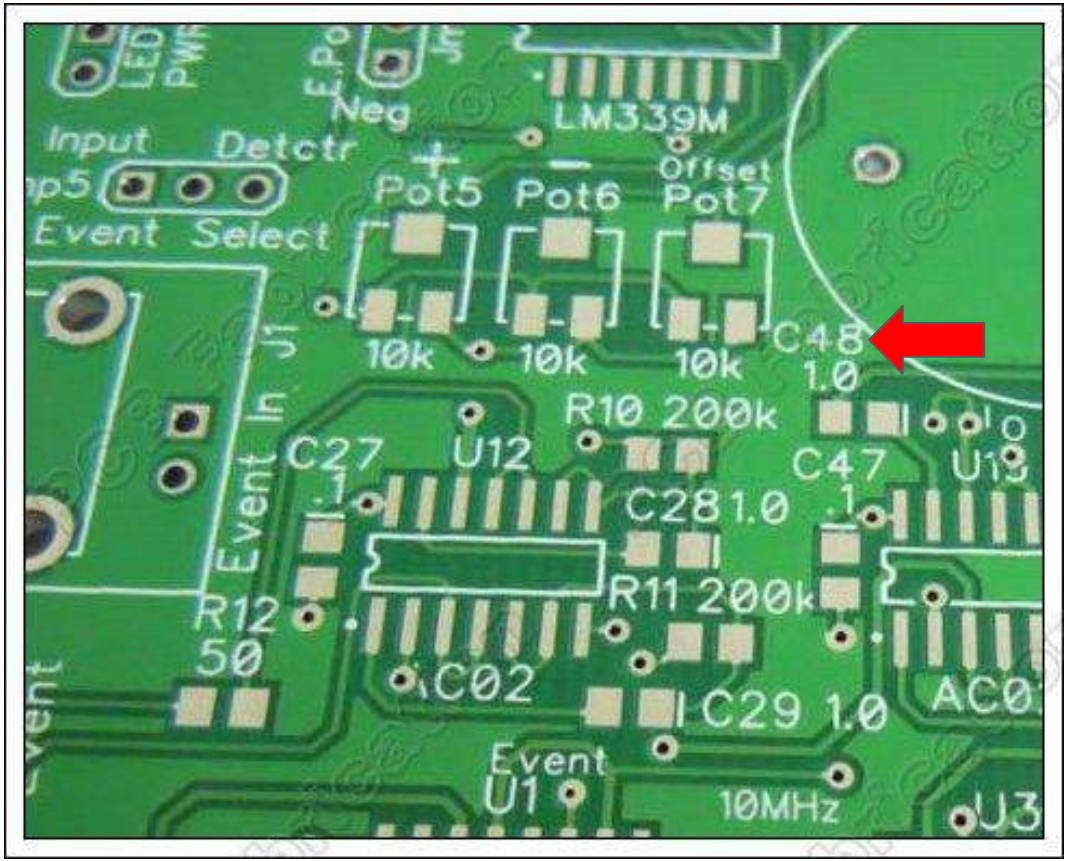
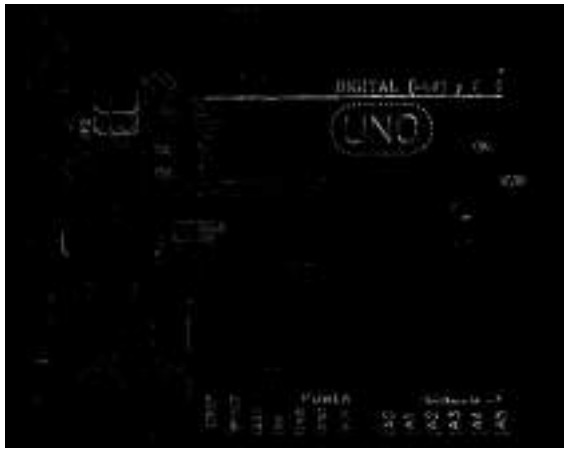


image by Nick John Eliopoulos

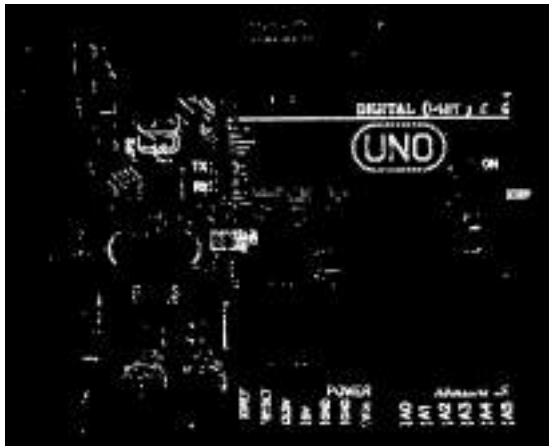


https://www.pcbdirectory.com/community/what-is-the-silkscreen-in-a-pcb

Input RGB Image

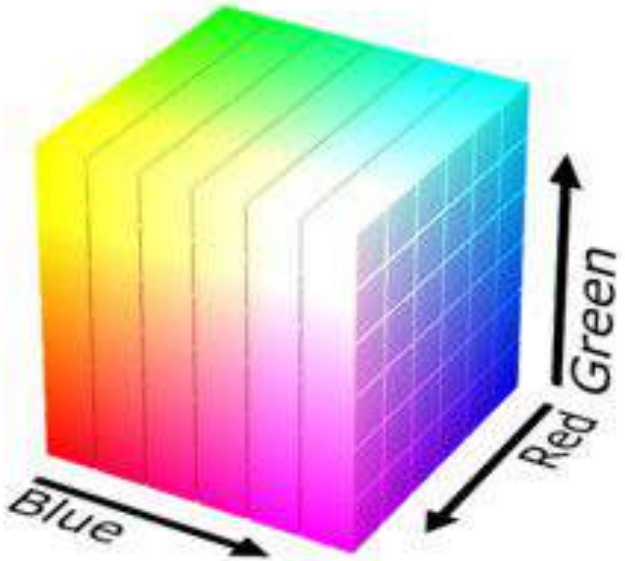


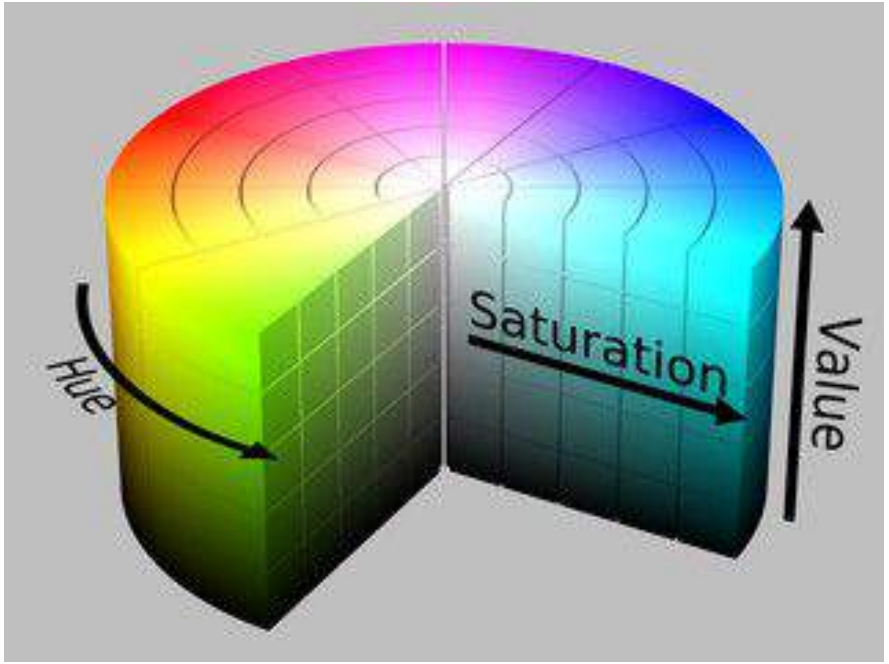
inRange



Eroded

HSV (hue, saturation, value) colorspace





https://docs.opencv.org/3.4/da/d97/tutorial_threshold_inRange.html

R' = R/255

G' = G/255

B' = B/255

Cmax = max(R', G', B')

Cmin = min(R', G', B')

Δ = Cmax - Cmin

Hue calculation:

$$H = \begin{cases} 60° \times \left(\frac{G'-B'}{\Delta} mod 6\right) & , Cmax = R' \\ 60° \times \left(\frac{B'-R'}{\Delta} + 2\right) & , Cmax = G' \\ 60° \times \left(\frac{R'-G'}{\Delta} + 4\right) & , Cmax = B' \end{cases}$$

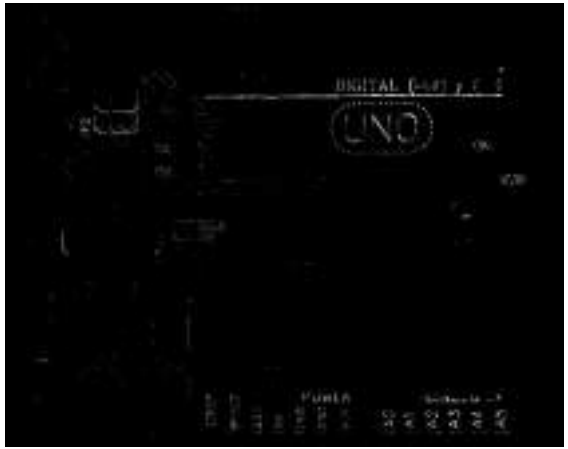
Saturation calculation:

$$S = \begin{cases} 0 & , C_{max} = 0 \\ \frac{\Delta}{C_{max}} & , C_{max} \neq 0 \end{cases}$$

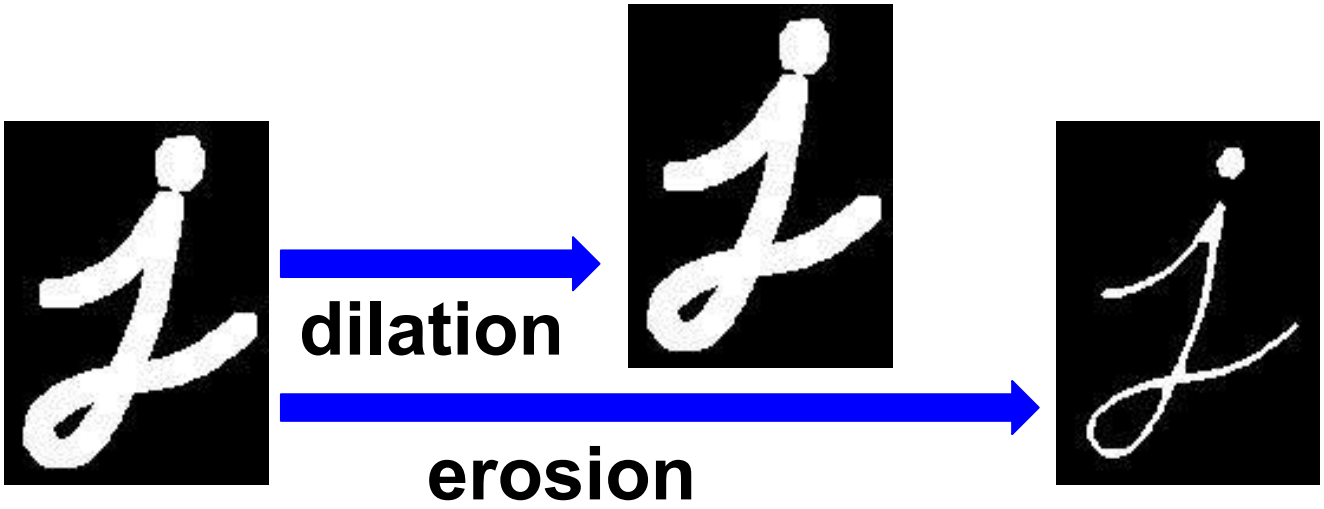
Value calculation: V = Cmax

https://math.stackexchange.com/questions/556341/rgb-to-hsv-color-conversion-algorithm

```
### Apply a threshold to the image
### These values were determined by sampling the pixel values of the silkscreen
lowerBoundRGBValues = (210,210,210)
upperBoundRGBValues = (230,230,230)
postThresholdImage = cv2.inRange(imageToSegment, lowerBoundRGBValues, upperBoundRGBValues)
```

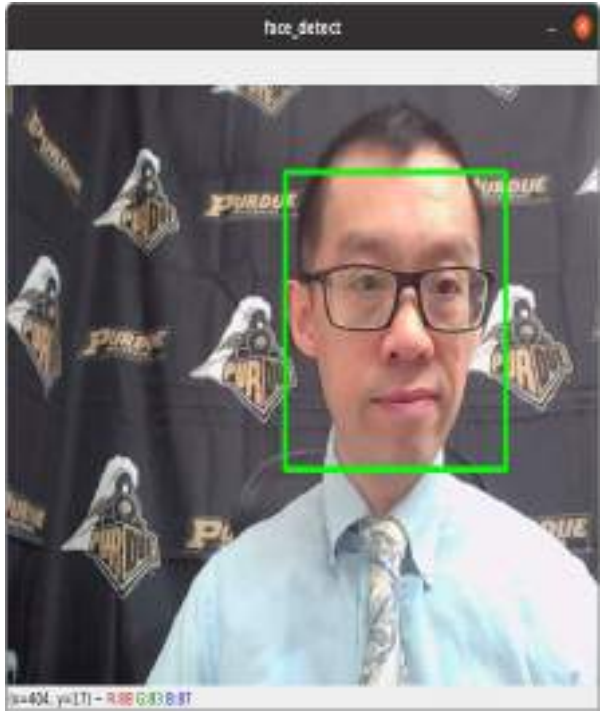


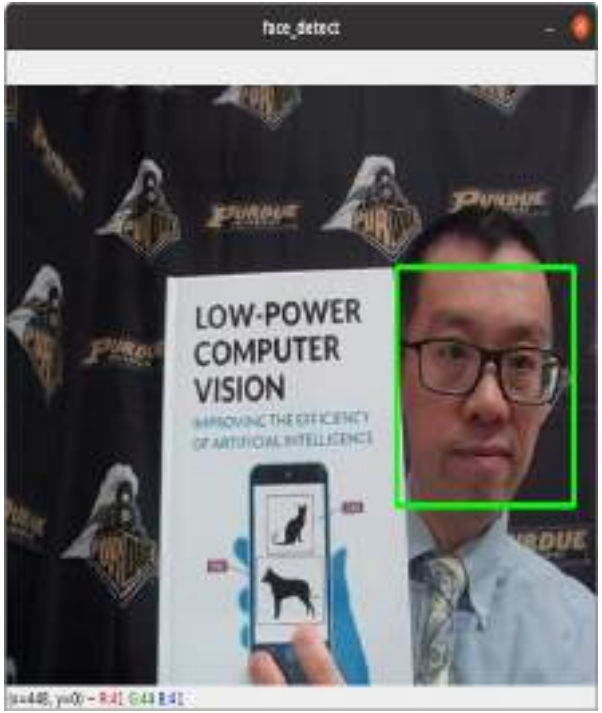
Dilation and Erosion (Morphological Operations)

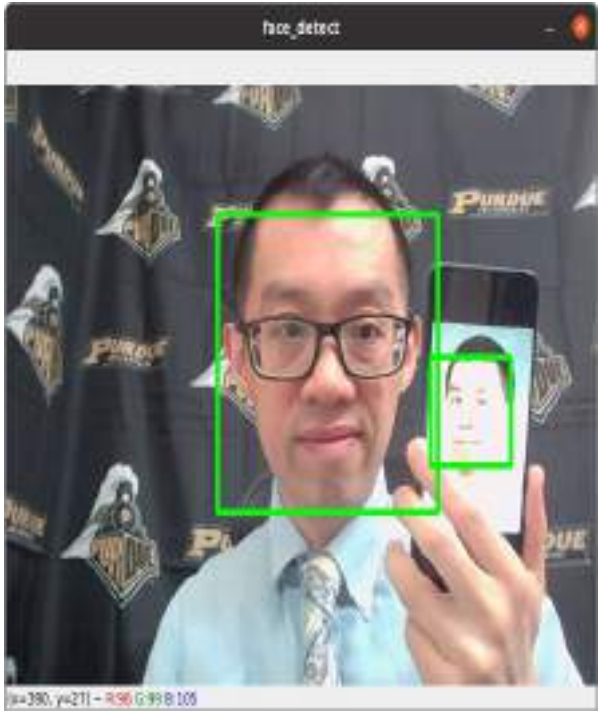


https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html

Detect Face







https://github.com/luyunghsiang/embeddedvision

```
import numpy, cv2, sys
imcap = cv2.VideoCapture(0)
imcap.set(3, 640) # set width as 640
imcap.set(4, 480) # set height as 480

faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

stop = False
while (stop == False):
    success, img = imcap.read() # capture frame from video
    # converting image from color to grayscale

    imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # Getting corners around the face
    # 1.3 = scale factor, 5 = minimum neighbor can be detected

    faces = faceCascade.detectMultiScale(imgGray, 1.3, 5)
    # drawing bounding box around face

    for (x, y, w, h) in faces:
        img = cv2.rectangle(img, (x, y), (x + w, y + h),
                            (0, 255,   0), 3)
        # displaying image with bounding box

        cv2.imshow('face_detect', img)

        # loop will be broken when 'q' is pressed on the keyboard
        if (cv2.waitKey(10) & 0xFF) == ord('q'):
            stop = True
cap.release()
cv2.destroyWindow('face_detect')
```