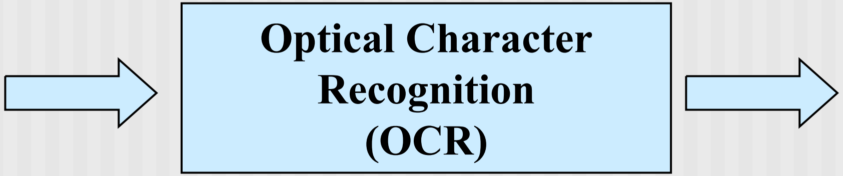
Lecture 2

Pattern Classification

Lecture by: Suthep “Jogie” Madarasmi, Ph.D.

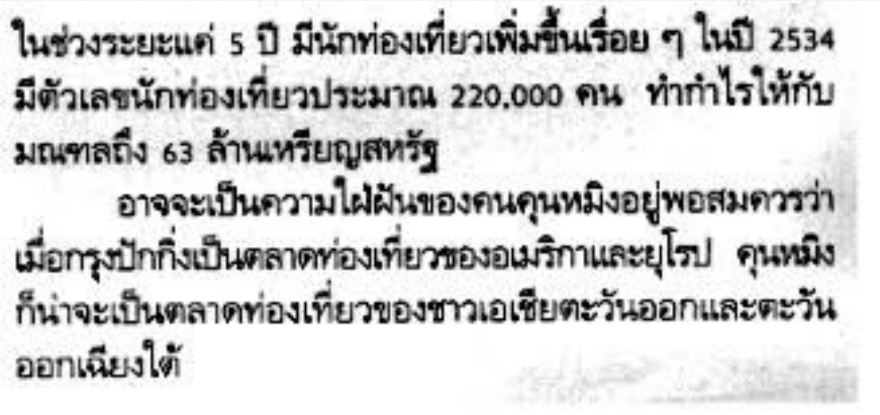
Optical Character Recognition OCR

Input: Scanned
Image of paper
with printed text



Optical Character Recognition (OCR)

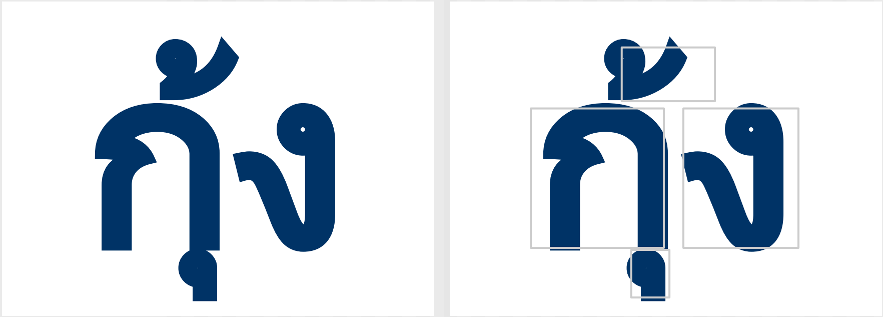
Output: Text
File in UTF-8
encoding

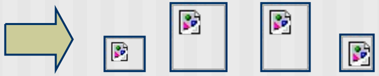


ในช่วงระยะแค่ 5 ปี มีนักท่องเที่ยวเพิ่มขึ้นเรื่อย ๆ ในปี 2534 มีตัวเลขนักท่องเที่ยวประมาณ 220,000 คน ทำกำไรให้กับมณฑลถึง 63 ล้านเหรียญสหรัฐ

อาจจะเป็นความใฝ่ฝันของคนคุนหมิงอยู่พอสมควรว่า เมื่อกรุงปักกิ่งเป็นตลาดท่องเที่ยวของอเมริกาและยุโรป คุนหมิงก็น่าจะเป็นตลาดท่องเที่ยวของชาวเอเชียตะวันออกและตะวันออกเฉียงใต้

segmentation - find the bounding box





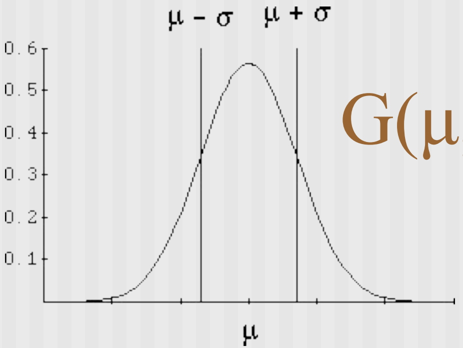
The 5 Steps in OCR Algorithm

1. **Remove image noise.** Use Gaussian Filtering and Median Filter.
2. **Threshold image to foreground/background.** Use Image Histogram and select threshold at valley between 2 peaks.
3. **Segmentation into single characters.** Use Contour Blob Coloring and get bounding box.
4. **Resize each test image to the template size.** Use aspect ratio to guide new image size.
5. **Character Recognition.** Use Template Matching (KNN), ANN, ML, …

Gaussian Additive Noise

- For example, an ideal white paper

220	220	220	220	220
220	220	220	220	220
220	220	220	220	220
220	220	220	220	220

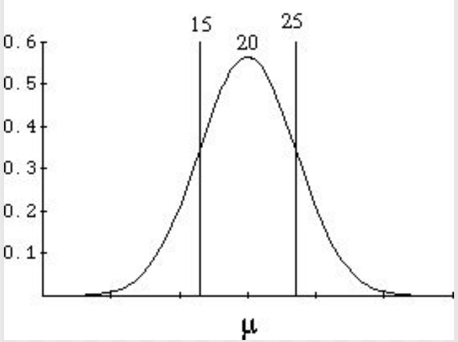


G(μ,σ)

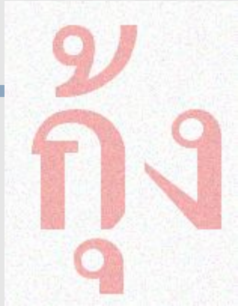
- Scanned white paper will spread around 220 + 20:

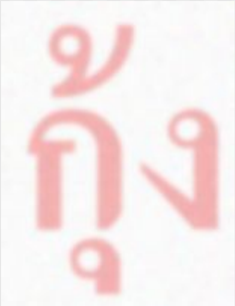
242	239	240	220	238
242	240	240	241	245
247	237	230	235	238
244	240	237	232	243

G(μ=20,σ = 5)

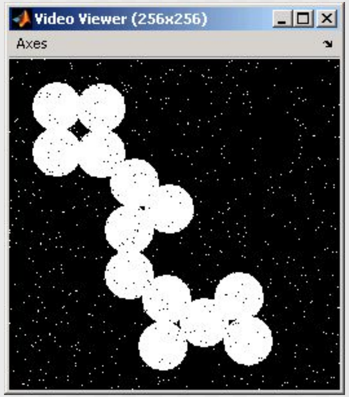


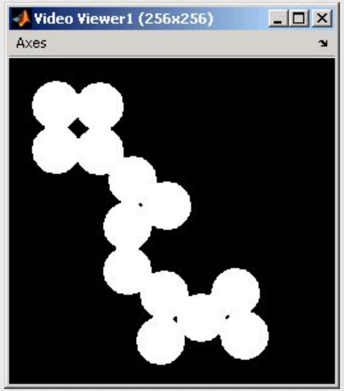
The Convolution Filter





- Gaussian Noise
  - Use Gaussian Filter
  - Can also use the Mean (Average) Filter



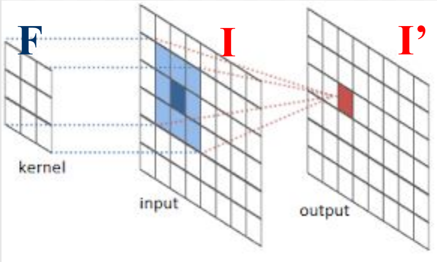


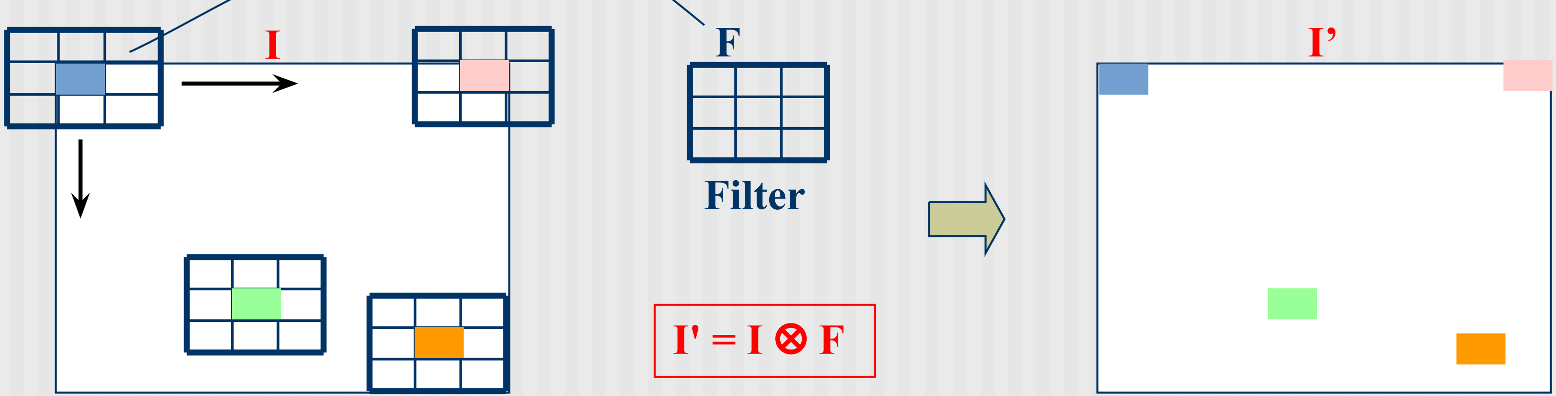
- Salt/Pepper Noise
  - Use Median Filter

The Convolution Filter

Synonyms for Convolution Matrix

- Kernel
- Filter
- Mask
- Template





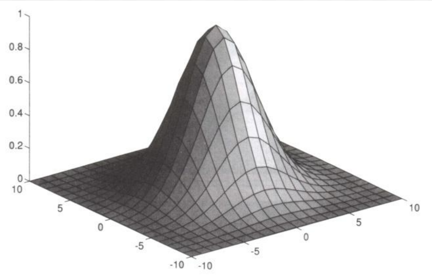
I' = I ⊗ F

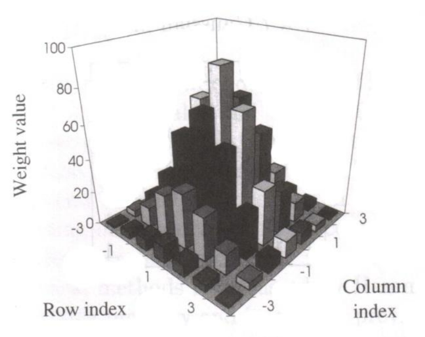
Input Image

Convolution
- Image Dot Product

Output Image

The Gaussian Filter (Convolution)





1	1	2	2	2	1	1
1	2	2	4	2	2	1
2	2	4	8	4	2	2
2	4	8	16	8	4	2
2	2	4	8	4	2	2
1	2	2	4	2	2	1
1	1	2	2	2	1	1

Step 1. Remove Image Noise

Median Filter: Rids Salt/Pepper Noise

- 20, 21, 32, 23, 17, 19 , 20
- Order by value

  17
  19
  20
  20
  21
  23
  32

Use the middle

- Use 20

- 20, 22, 32, 23, 17, 19 , 20,24
- Order by value

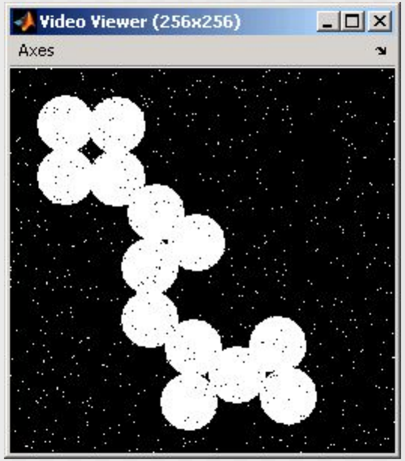
  17
  19
  20
  20
  22
  23
  24
  32

Average the middle
(20+22)/2 = 21

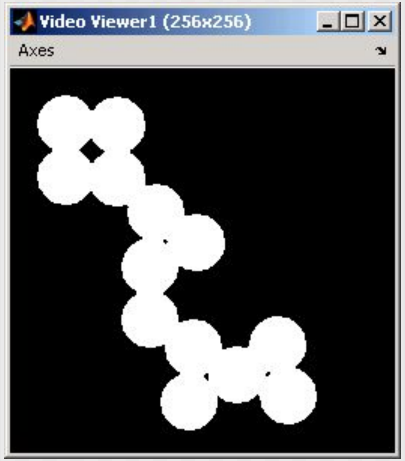
- Use 21

Step 1. Remove Image Noise

Salt/Pepper Noise Removal



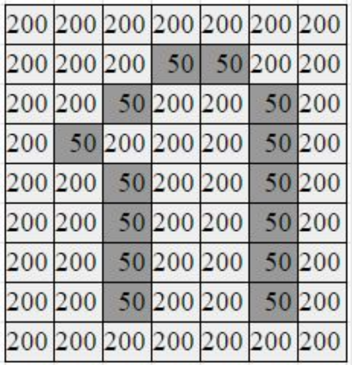
Noise Removal



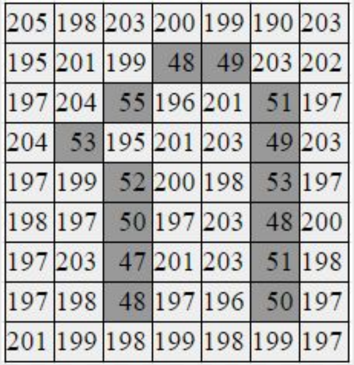
The 5 Steps in OCR Algorithm

1. **Remove image noise.** Use Gaussian Filtering and Median Filter.
2. **Threshold image to foreground/background.** Use Image Histogram and select threshold at valley between 2 peaks.
3. **Segmentation into single characters.** Use Contour Blob Coloring and get bounding box.
4. **Resize each test image to the template size.** Use aspect ratio to guide new image size.
5. **Character Recognition.** Use Template Matching (KNN), ANN, ML, …

2 color images have shades of gray from Gaussian noise



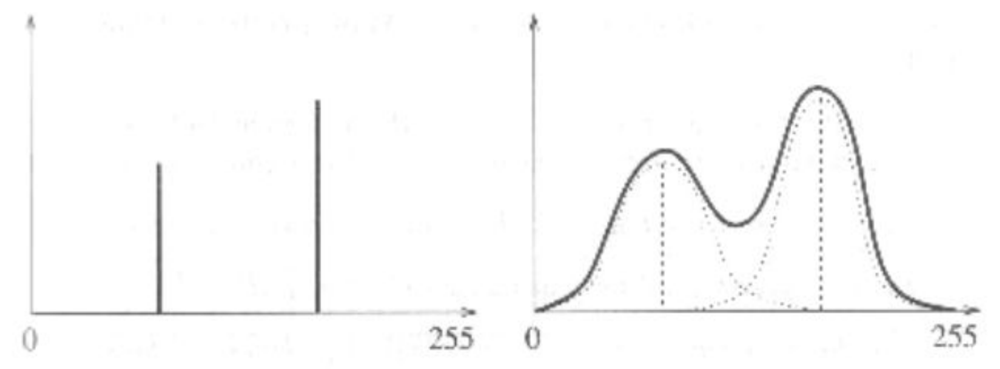
200	200	200	200	200	200	200
200	200	200	50	50	200	200
200	200	50	200	200	50	200
200	50	200	200	200	50	200
200	200	50	200	200	50	200
200	200	50	200	200	50	200
200	200	50	200	200	50	200
200	200	50	200	200	50	200
200	200	200	200	200	200	200



205	198	203	200	199	190	203
195	201	199	48	49	203	202
197	204	55	196	201	51	197
204	53	195	201	203	49	203
197	199	52	200	198	53	197
198	197	50	197	203	48	200
197	203	47	201	203	51	198
197	198	48	197	196	50	197
201	199	198	199	198	199	197

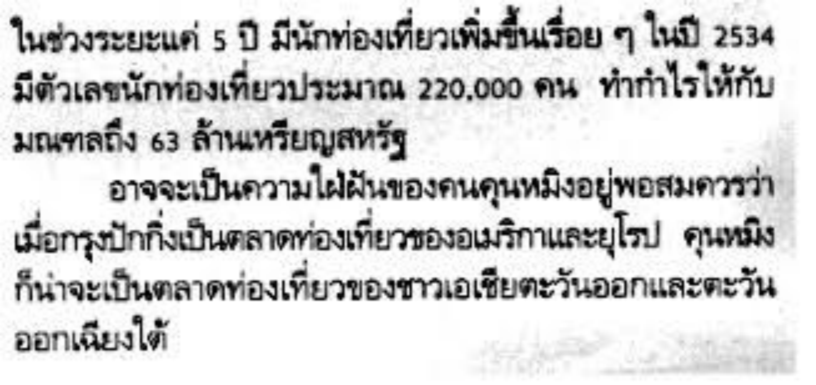
IDEAL Binary Image

Actual Image with Gaussian Noise



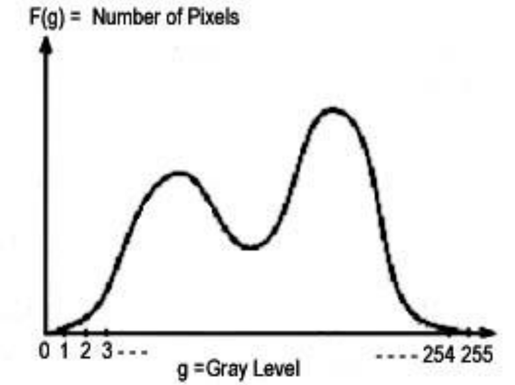
Histogram: Good summary of an Image

- g - gray level 0..255. (Use values or luminance channel, but can be each of RGB)
- F(g) - Represents frequency of gray level (g)
- F(1) - how many 1s (black) are there in image
- F(100) – how many 100 (gray) are there in image



ในช่วงระยะแค่ 5 ปี มีนักท่องเที่ยวเพิ่มขึ้นเรื่อย ๆ ในปี 2534
มีตัวเลขนักท่องเที่ยวประมาณ 220.000 คน ทำกำไรให้กับ
มณฑลถึง 63 ล้านเหรียญสหรัฐ

อาจจะเป็นความใฝ่ฝันของคนคุนหมิงอยู่พอสมควรว่า
เมื่อกรุงปักกิ่งเป็นตลาดท่องเที่ยวของอเมริกาและยุโรป คุนหมิง
ก็น่าจะเป็นตลาดท่องเที่ยวของชาวเอเชียตะวันออกและตะวัน
ออกเฉียงใต้

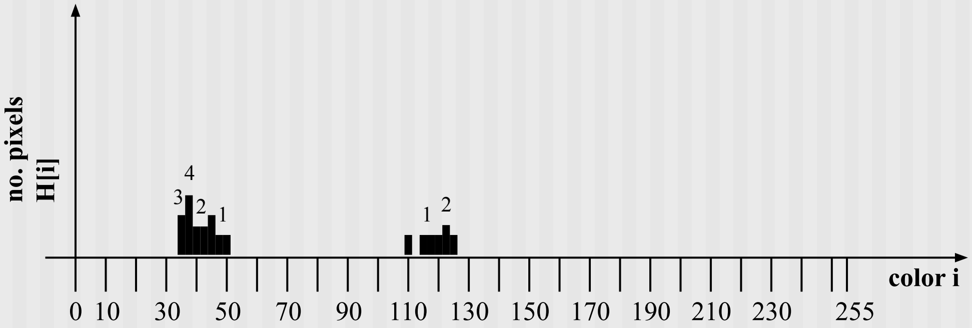


Step 2. Image Thresholding to get binary image

Histogram shows count of each color

Input Gray Image:

40	38	120	110	123
42	41	115	118	124
39	44	119	121	123
43	38	42	42	38
39	40	39	41	39



Histogram above Image:

[i]:	0	...	38	39	40	41	42	43	44	...	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	...	255
H [i]:	0	...	3	4	2	2	3	1	1	...	1	0	0	0	0	1	0	0	1	1	1	1	0	2	1	...	0

Note that the histogram of a gray scale image of any size can be represented by an integer array of 256 dimensions

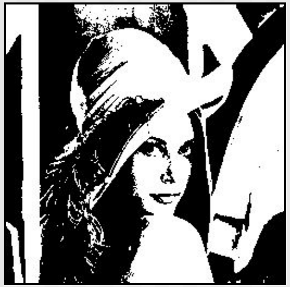
Histogram of 2 color image

- Generally looking for 2 peaks
- Look for the gray in between

Grayscale

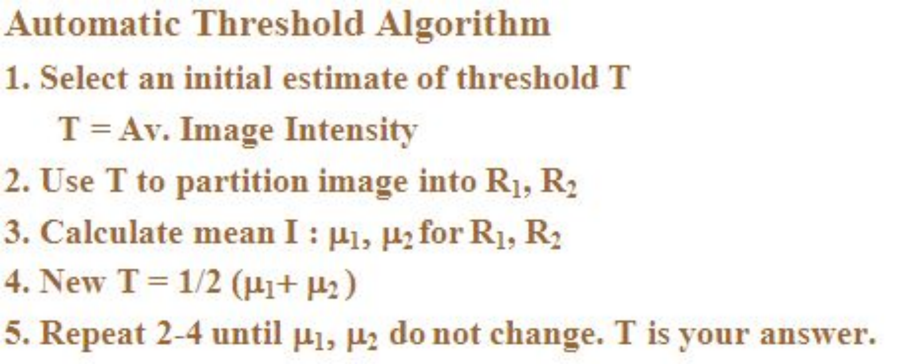
T = 128

Black/White



Step 2. Image Thresholding to get binary image

Automatic Threshold Example



Automatic Threshold Algorithm

1. Select an initial estimate of threshold T

   T = Av. Image Intensity

2. Use T to partition image into R_1, R_2

3. Calculate mean I : μ_1, μ_2 for R_1, R_2

4. New T = 1/2 (μ_1+ μ_2)

5. Repeat 2-4 until μ_1, μ_2 do not change. T is your answer.

Input Gray Image:

40	38	120	110	123
42	41	115	118	124
39	44	119	121	123
43	38	42	42	38
39	40	39	41	39

Output Binary Image:

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

T_1 = 69.72 = 70
R1_1 <= 70
R2_1 > 70

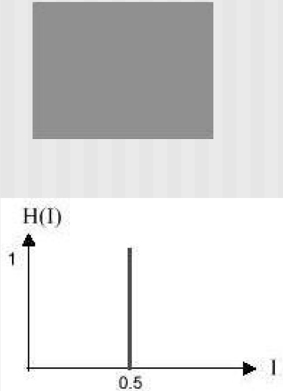
μ_R1 = 40
μ_R2 = 119
T_2 = 80

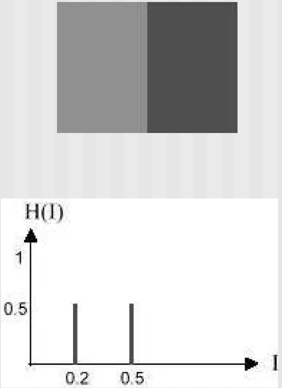
R1_2 <= 80
R2_2 > 80
μ_R1 = 40
μ_R2 = 119
T_3 = 80

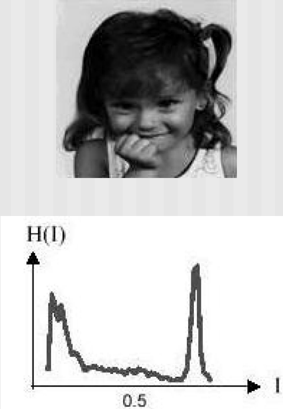
ANSWER: Threshold = 80.

Histogram Examples

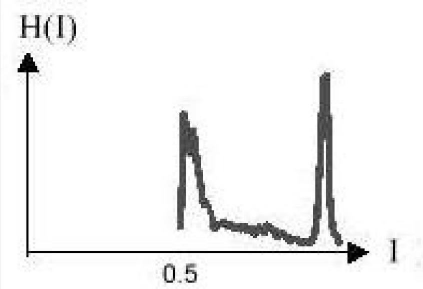
- Represent the number of pixels according to gray levels

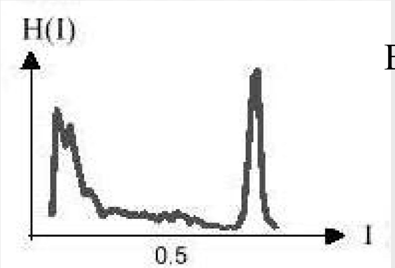






Modification of histogram





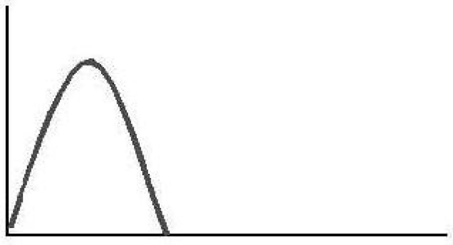
new = (old - min) / (max - min)

Example:

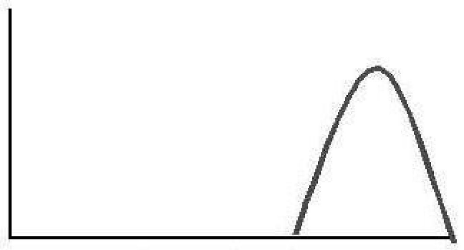
new = (old - 0.5) / (1 - 0.5)

Brightness

- Brightness is the average pixel value



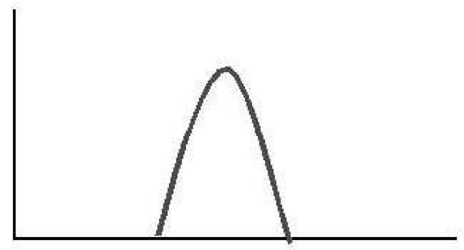
− Brightness



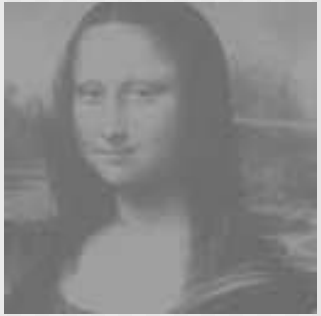
+ Brightness

Contrast

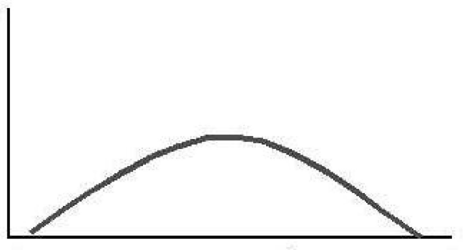
- relative differences in image



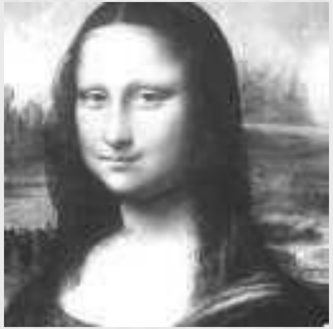
− Contrast



new = (old - min) / (max - min)



+ Contrast



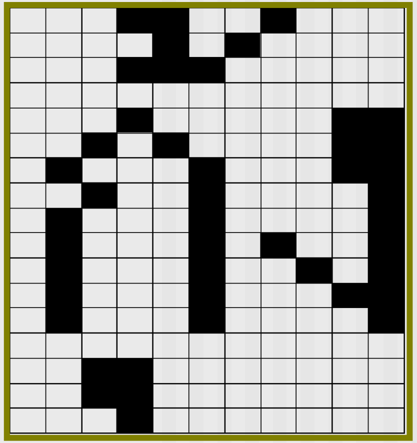
The 5 Steps in OCR Algorithm

1. **Remove image noise.** Use Gaussian Filtering and Median Filter.
2. **Threshold image to foreground/background.** Use Image Histogram and select threshold at valley between 2 peaks.
3. **Segmentation into single characters.** Use Contour Blob Coloring and get bounding box.
4. **Resize each test image to the template size.** Use aspect ratio to guide new image size.
5. **Character Recognition.** Use Template Matching (KNN), ANN, ML, …

The 5 Steps in OCR Algorithm

1. **Remove image noise.** Use Gaussian Filtering and Median Filter.
2. **Threshold image to foreground/background.** Use Image Histogram and select threshold at valley between 2 peaks.
3. **Segmentation into single characters.** Use Contour Blob Coloring and get bounding box.
4. **Resize each test image to the template size.** Use aspect ratio to guide new image size.
5. **Character Recognition.** Use Template Matching (KNN), ANN, ML, …

Character Extraction via Connected Components



			1	1			2			
				1		2				
			1	1	1					
			3						4	4
		3		3					4	4
	3				3				4	4
		3			3					4
	3				3					4
	3				3		5			4
	3				3			5		4
	3				3				5	4
	3				3					4
		6	6							
		6	6							
			6							

Equivalence table:

1	2
4	5

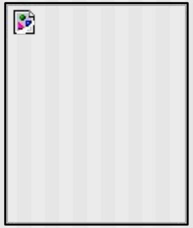
Template for Contour Coloring:

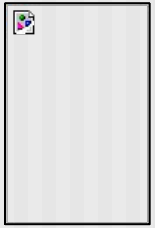
NW	N	NE
W	o	

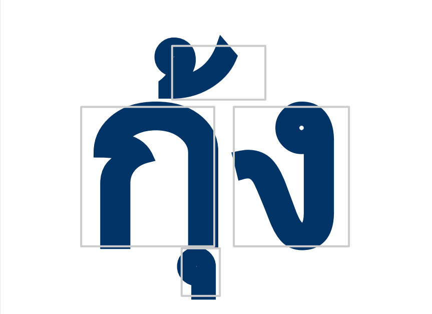
Template for Region Coloring:

	N
W	o

Characters extracted using same color bounding box:





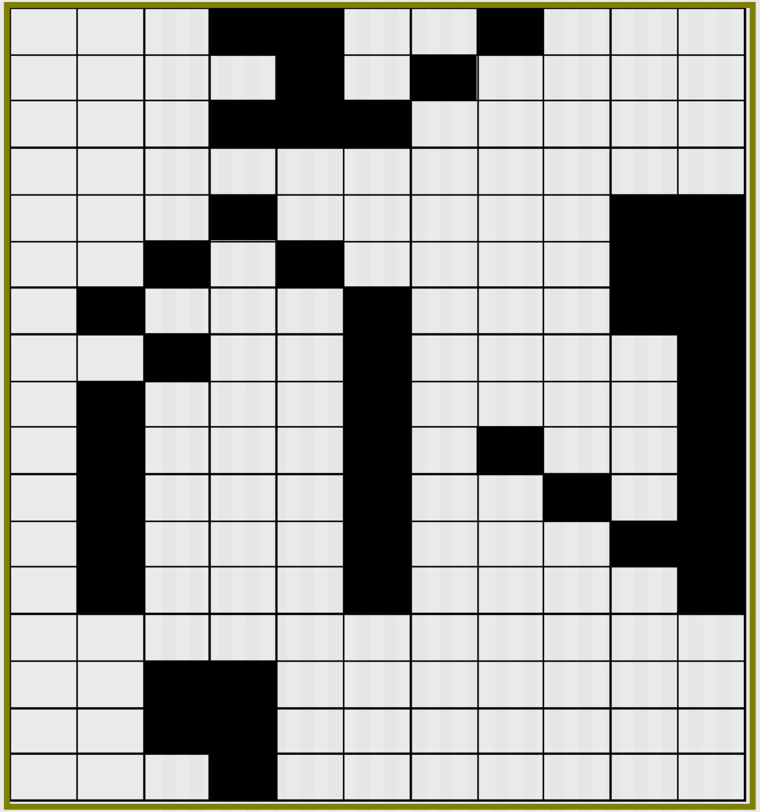


Step 3. Character Segmentation by Contour Coloring

Let’s Try

NW	N	NE
W	o	

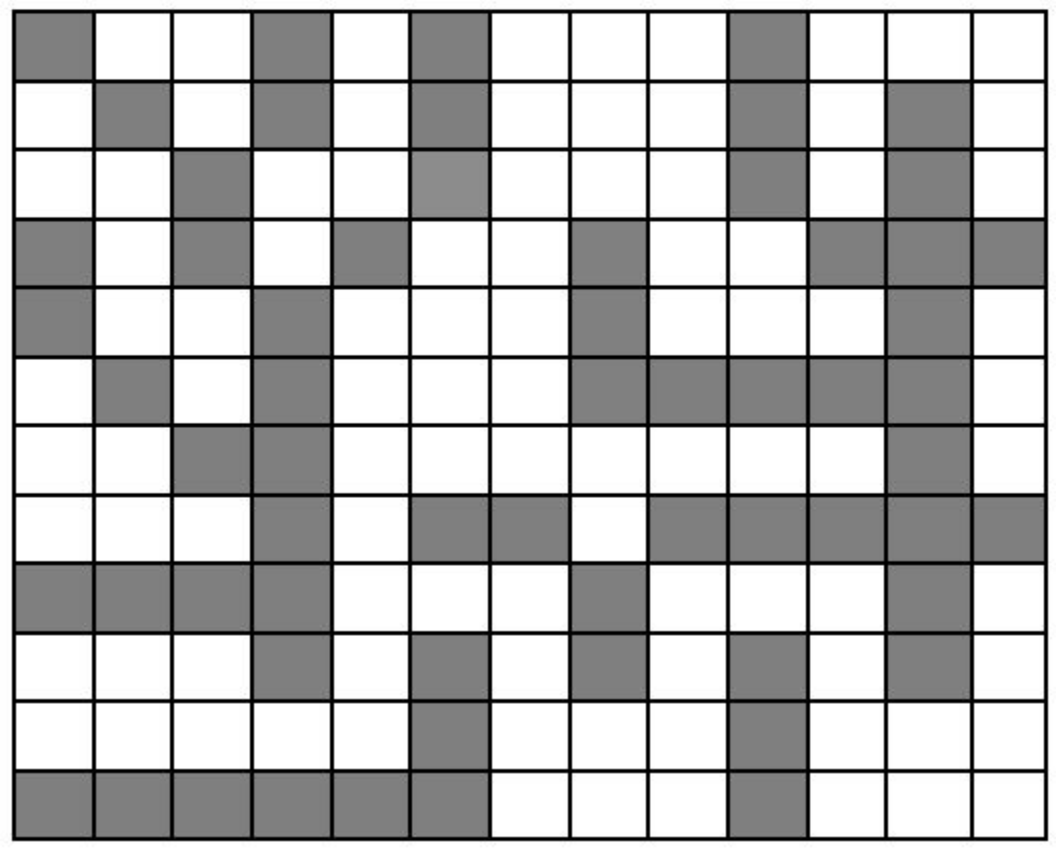
	N
W	o



Let’s Try

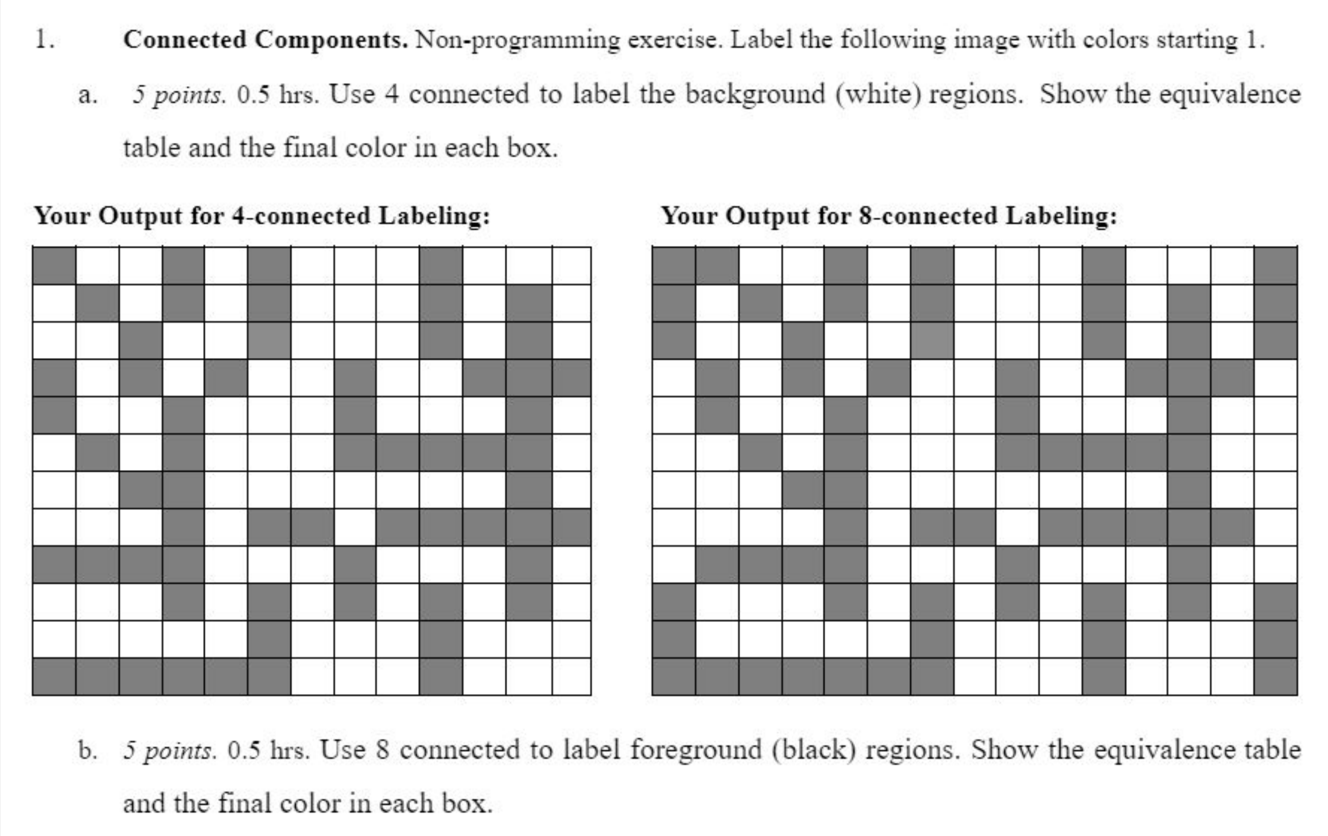
NW	N	NE
W	o	

	N
W	o



■	□	□	■	□	■	□	□	□	■	□	□	□
□	■	□	■	□	■	□	□	□	■	□	■	□
□	□	■	□	□	■	□	□	□	■	□	■	□
■	□	■	□	■	□	□	■	□	□	■	■	■
■	□	□	■	□	□	□	■	□	□	□	■	□
□	■	□	■	□	□	□	■	■	■	■	■	□
□	□	■	■	□	□	□	□	□	□	□	■	□
□	□	□	■	□	■	■	□	■	■	■	■	■
■	■	■	■	□	□	□	■	□	□	□	■	□
□	□	□	■	□	■	□	■	□	■	□	■	□
□	□	□	□	□	■	□	□	□	■	□	□	□
■	■	■	■	■	■	□	□	□	■	□	□	□

Quiz 3



1. **Connected Components.** Non-programming exercise. Label the following image with colors starting 1.

   a. *5 points.* 0.5 hrs. Use 4 connected to label the background (white) regions. Show the equivalence table and the final color in each box.

Your Output for 4-connected Labeling:

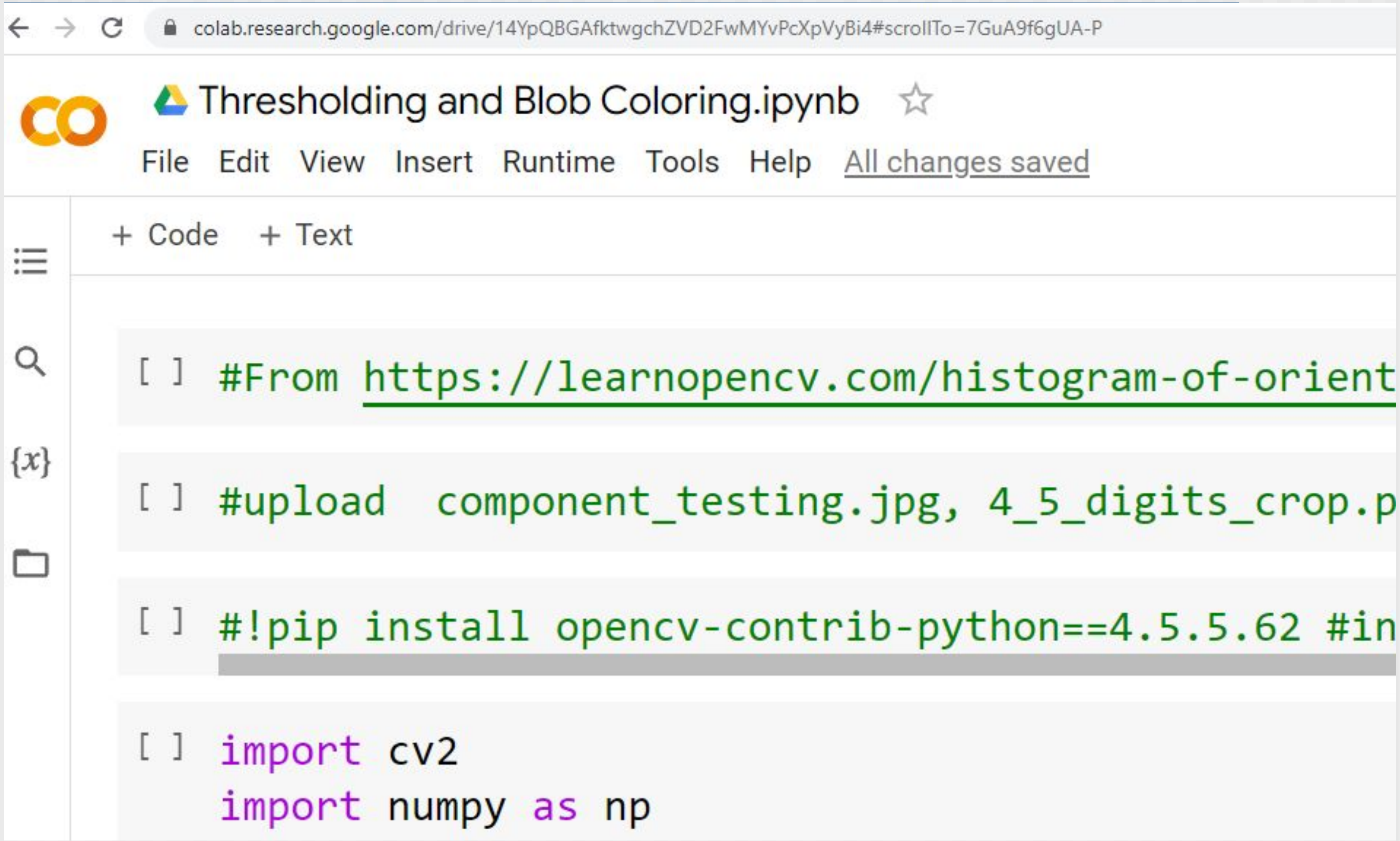
■			■		■				■			
	■		■		■				■		■	
		■			■				■		■	
■		■		■			■			■	■	■
■			■				■				■	
	■		■				■	■	■	■	■	
		■	■								■	
			■		■	■		■	■	■	■	■
■	■	■	■				■				■	
			■		■		■		■		■	
					■				■			
■	■	■	■	■	■				■			

Your Output for 8-connected Labeling:

■	■			■		■				■				■
■		■		■		■				■		■		■
■			■			■				■		■		■
	■		■		■			■			■	■	■	
	■			■				■				■		
		■		■				■	■	■	■	■		
			■	■								■		
				■		■	■		■	■	■	■	■	
	■	■	■	■				■				■		
■				■		■		■		■		■		■
■						■				■				■
■	■	■	■	■	■	■				■				■

   b. *5 points.* 0.5 hrs. Use 8 connected to label foreground (black) regions. Show the equivalence table and the final color in each box.

Learning Exercise on Thresholding, Connected Components


colab.research.google.com/drive/14YpQBGAfktwgchZVD2FwMYvPcXpVyBi4#scrollTo=7GuA9f6gUA-P
Thresholding and Blob Coloring.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
#From https://learnopencv.com/histogram-of-orient
```

```
#upload  component_testing.jpg, 4_5_digits_crop.p
```

```
#!pip install opencv-contrib-python==4.5.5.62 #in
```

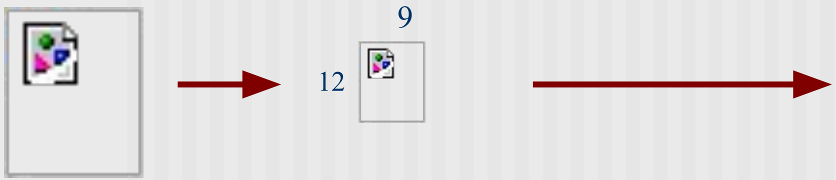
```
import cv2
import numpy as np
```

The 5 Steps in OCR Algorithm

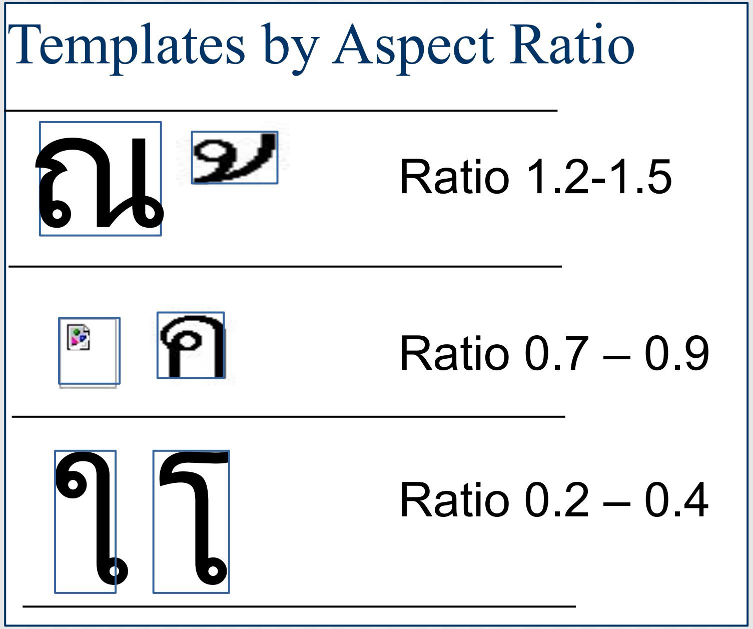
1. **Remove image noise.** Use Gaussian Filtering and Median Filter.
2. **Threshold image to foreground/background.** Use Image Histogram and select threshold at valley between 2 peaks.
3. **Segmentation into single characters.** Use Contour Blob Coloring and get bounding box.
4. **Resize each test image to the template size.** Use aspect ratio to guide new image size.
5. **Character Recognition.** Use Template Matching (KNN), ANN, ML, …

Image scaling needed for OCR

Resize based on aspect ratio width/height

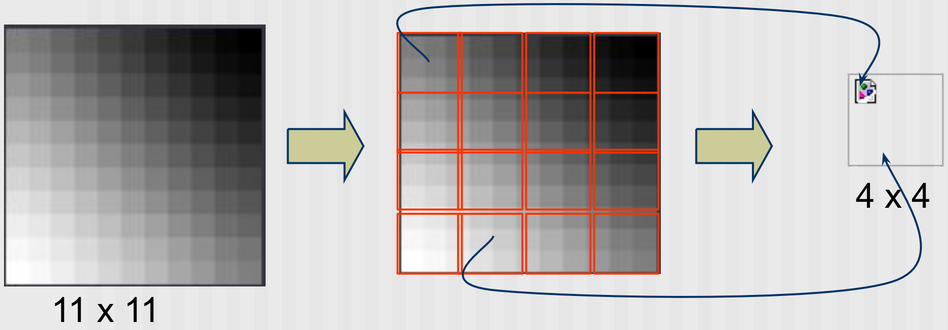


Template Matching within the same aspect ratio group.


Templates by Aspect Ratio

Ratio 1.2-1.5
Ratio 0.7 – 0.9
Ratio 0.2 – 0.4

How to reduce image size (raster images)?



Box Size = 11/4 = (2.75x, 2.75y)

Use Grayscale color average weighted by area of window.

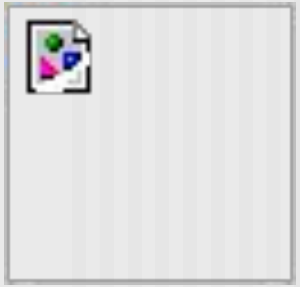
Use Box Sampling to Reduce Image Size:

Consider above example of reducing input 11 x 11 image to an output 4 x 4 image. You can do this by taking the 11 x 11 image and overlaying it with a 4 x 4 tile as shown in middle image above. That means each output tile will take up about 2.75 pixels of the input. So, you can do a weighted average of the input based on pixel area covered to get the output. Thus, the (row, column) pixel output I(1,1) will come from input's weighted average based on areas occupied by the input pixel

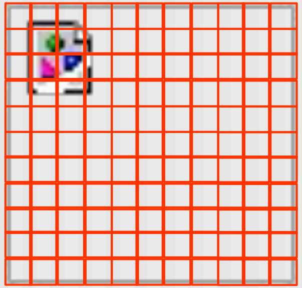
$$I_{new}(1,1) = \frac{1*1*I(1,1)+1*1*I(1,2)+1*0.75*I(1,3)+1*1*I(2,1)+1*1*I(2,2)+1*0.75*I(2,3)+0.75*1*I(3,1)+0.75*1*I(3,2)+0.75*0.75*I(3,3)}{1+1+0.75+1+1+0.75+0.75+0.75+0.75*0.75}$$

The output's I(1, 2) will have influence from 16 pixels because the box (1,2) touches 16 pixels. And so on.

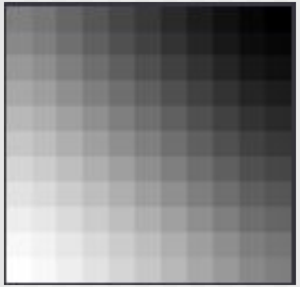
How to increase image size?



4 x 4



11 x 11



11 x 11

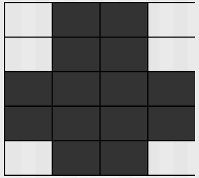
Box Size = 11/4 = (2.75x, 2.75y)

The 5 Steps in OCR Algorithm

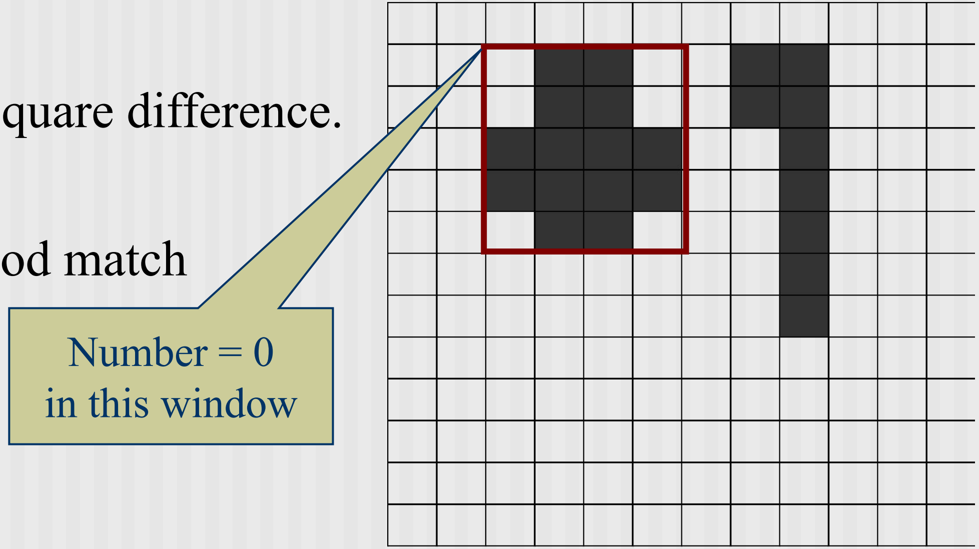
1. **Remove image noise.** Use Gaussian Filtering and Median Filter.
2. **Threshold image to foreground/background.** Use Image Histogram and select threshold at valley between 2 peaks.
3. **Segmentation into single characters.** Use Contour Blob Coloring and get bounding box.
4. **Resize each test image to the template size.** Use aspect ratio to guide new image size.
5. **Character Recognition.** Use Template Matching (KNN), ANN, ML, …

Template matching to image

- Given this template:



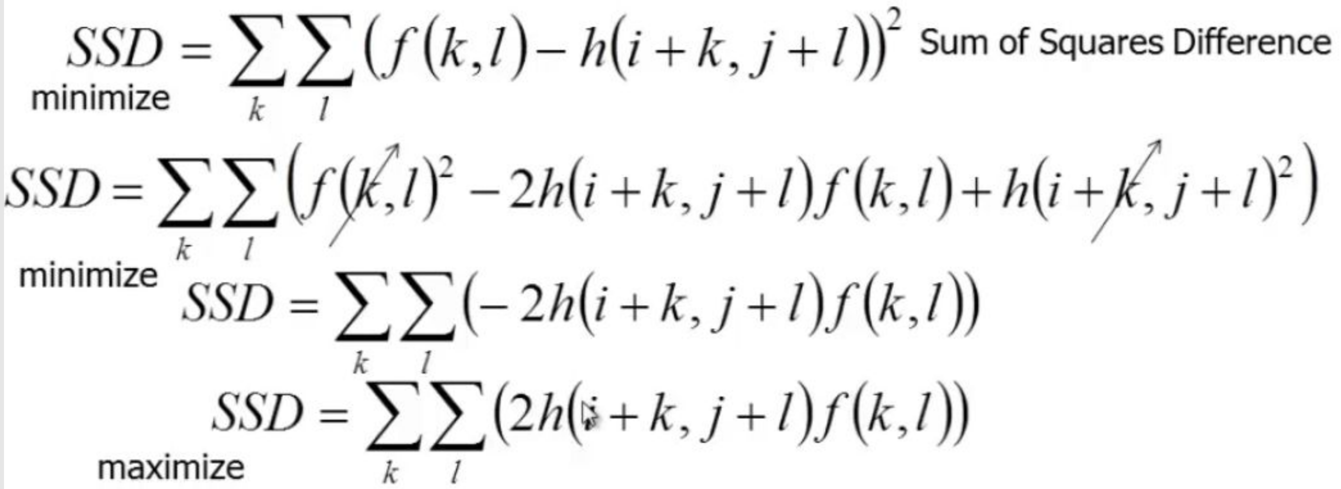
- Place template, do sum square difference.
- Move throughout image.
- Low number means a good match



Number = 0
in this window

Step 5. Character recognition (template match, NN, ML, …)

Min Template Match = Max Convolution



$$SSD = \sum_{k}\sum_{l}\left(f(k,l) - h(i+k, j+l)\right)^2$$ Sum of Squares Difference

minimize

$$SSD = \sum_{k}\sum_{l}\left(f(\cancel{k},l)^2 - 2h(i+k, j+l)f(k,l) + h(i+\cancel{k}, j+l)^2\right)$$

minimize

$$SSD = \sum_{k}\sum_{l}\left(-2h(i+k, j+l)f(k,l)\right)$$

$$SSD = \sum_{k}\sum_{l}\left(2h(i+k, j+l)f(k,l)\right)$$

maximize

f(*k,l*) is template image. h(*i, j*) is input image.

Template matching is done at h(*i, j*).

Range of *k* is template height, range of *l* is image width.

Min Template Match = Max Convolution

$$\underset{\text{minimize}}{SSD} = \sum_{k}\sum_{l}\left(f(k,l) - h(i+k, j+l)\right)^2$$ Sum of Squares Difference

f(i,j):

	■	■	
	■	■	
■	■	■	■
■	■	■	■
	■	■	

h(i,j):

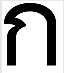
			■	■			■	■			
			■	■			■	■			
		■	■	■	■			■			
		■	■	■	■			■			
			■	■				■			
								■			
								■			

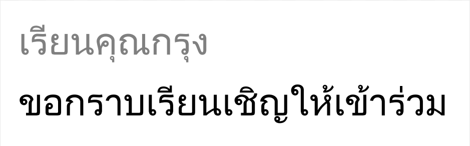
```
for i = 1 .. 13
for j = 1.. 12
  r(i,j) = 0
  for k = -2 .. 2
  for l = -2.. 2
     r(i,j) += (f(k, l) - h(i+k, j+l))^2
```

Image Invariant Features

Invariance to:

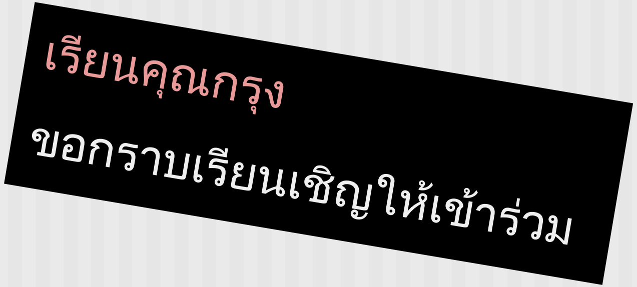
- Lighting
- Position
- Scale
- Orientation





เรียนคุณกรุง

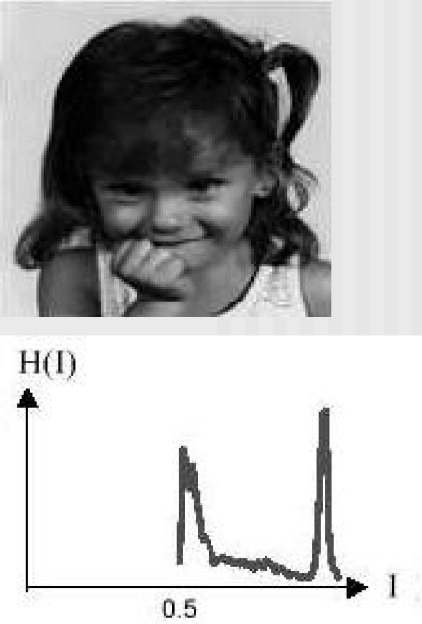
ขอกราบเรียนเชิญให้เข้าร่วม

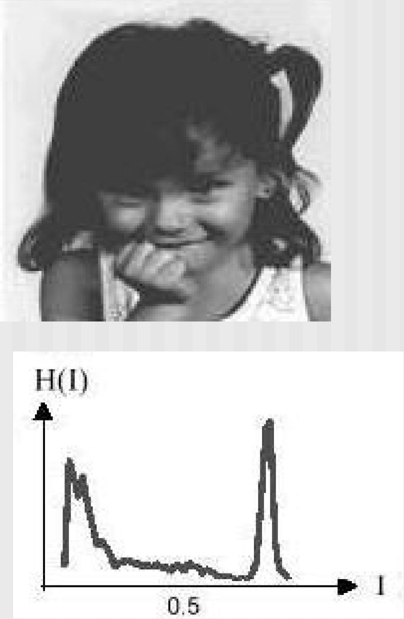


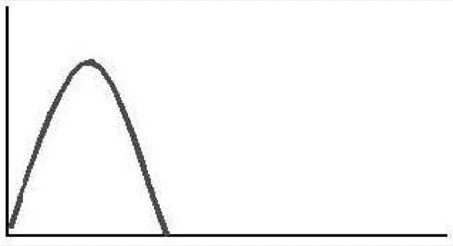
เรียนคุณกรุง

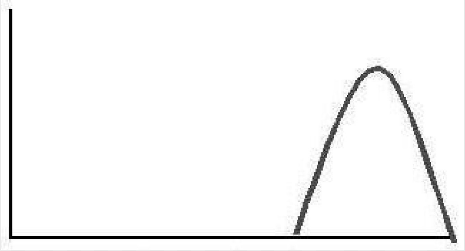
ขอกราบเรียนเชิญให้เข้าร่วม

Histograms are somewhat lighting invariant









− Brightness

+ Brightness

Color Invariant Features: Edges

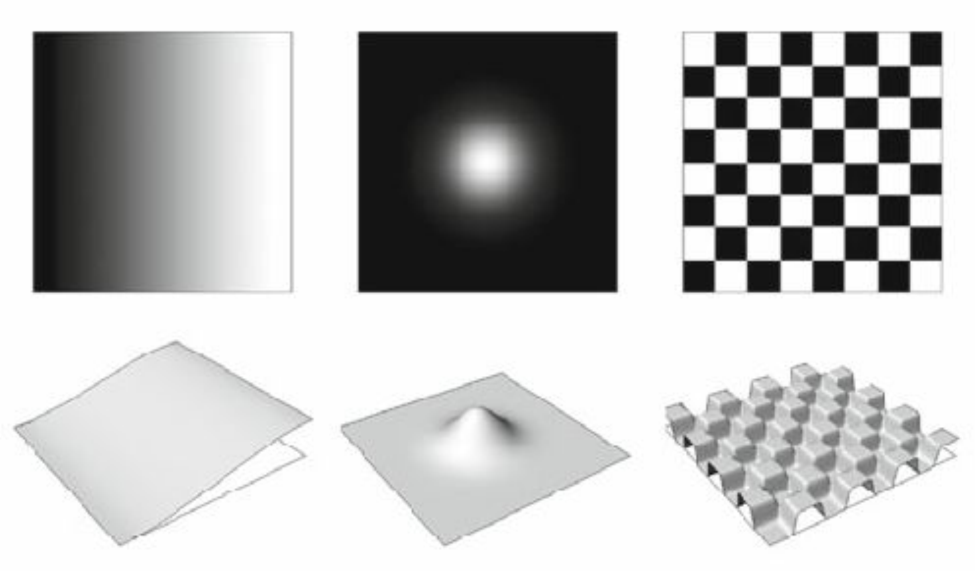
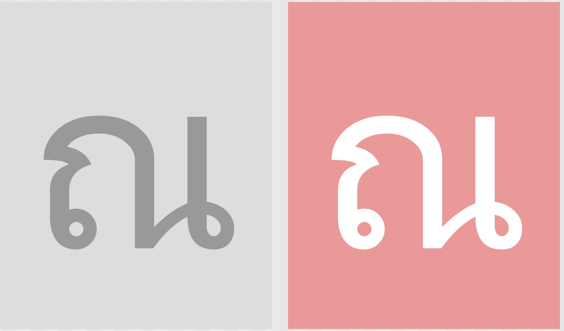


Image I(x,y) viewed as a surface plot Z(x, y)

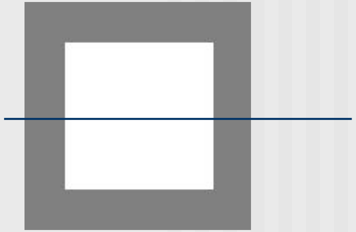


Same Edges

Edges will not get
you contour strokes

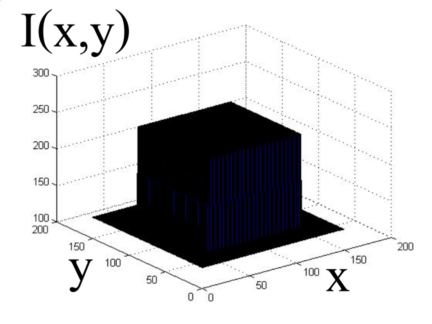
An Edge Point

Edge - Points in image with a big change in intensity



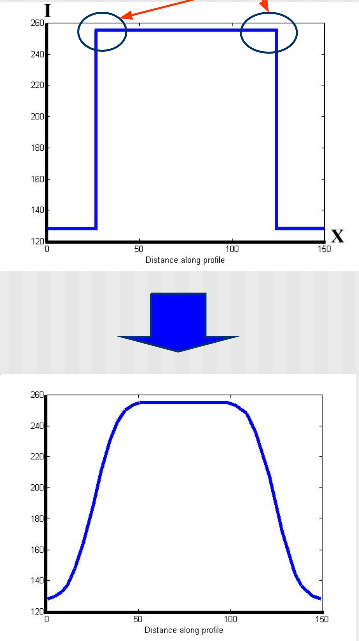
A Scan Line

Step Edge



Intensity Profile (Surface)

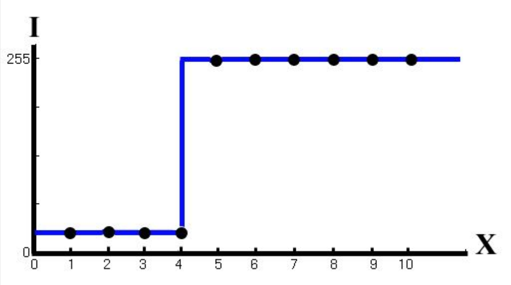
Edge Points



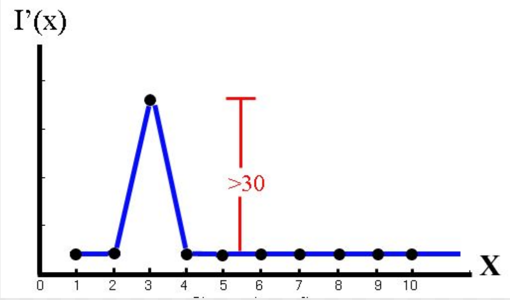
Step Edge Blurring
(Noise Reduction)

A simple Edge Detector

Edge if $\Delta I / \Delta X > 30$



The Input Image



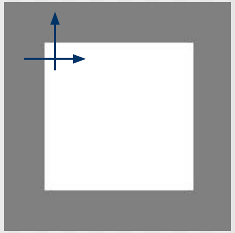
The First Difference Image

A step edge in X and Y

If $|dx| > T$ or $|dy| > T$ then edge. $T = 15$.

Problem when there is Gaussian noise, you can get undesired edges.

So, best to Gaussian blur before taking first difference



240	240	240	240	240
240	240	240	240	240
240	240	240	240	240
240	240	240	240	240

Just $\boldsymbol{\sigma} = 5$, can cause a problem

235	249	248	230	238
242	240	240	241	245
247	237	230	251	238
232	240	237	232	243

Use Gaussian Filter (Blur) to Reduce Noise

- We should blur the image first, because first difference in image with Gaussian noise can lead to a false edge. Consider

  I = 200 ± G(0, 20)

  Edge is when |dI/dx| ≥ 30. 4 edges due to noise!

I(x):	**170**	210	**220**	**180**	**210**	160
dI(x)/dx = $\Delta I = I_{x+1} - I_x$:	**40**	10	**-40**	**30**	**-50**	undefined
I_av(x):	200	200	203	203	183	183
dI_av(x)/dx	0	3	0	-20	0	undefined

Common Edge Masks: Prewitt

Prewitt Edge Operator

$G = G_x^{2} + G_y^{2} > T$ ➡ Edge

Gx

-1	0	1
-1	0	1
-1	0	1

Gy

1	1	1
0	0	0
-1	-1	1

I(x+1) - I(x-1)

Common Edge Detection: Sobel

Sobel Operator. Based on 1st derivative. Better because more blurring.

$$G = G_x{}^2 + G_y{}^2 > T$$ ➡ Edge

G_x

-1	0	1
-2	0	2
-1	0	1

G_y

1	2	1
0	0	0
-1	-2	-1

Gradient Magnitude G and Direction θ

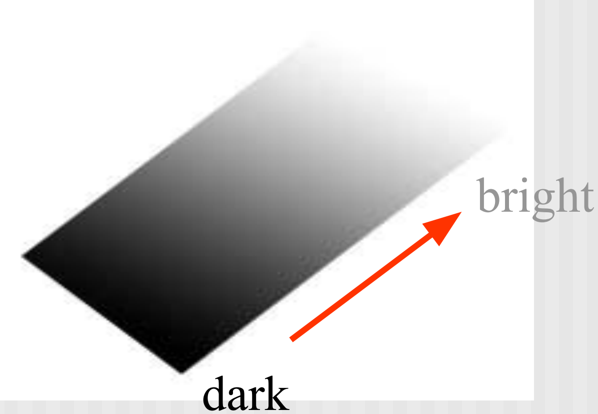
Gradient Direction is perpendicular to edge direction, from dark to bright.

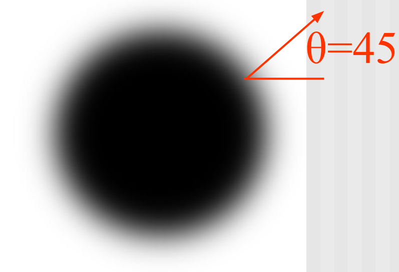
$$\tan\theta = G_y \;/\; G_x$$

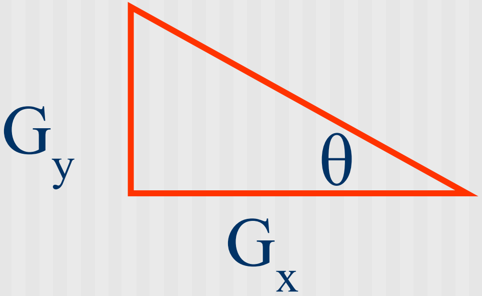
$$\theta = \arctan(G_y \;/\; G_x)$$

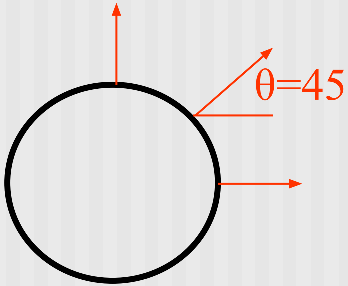
$$G^2 = G_x^{\;2} + G_y^{\;2}$$

In python use **atan2**
to get -180 to 180 degrees.



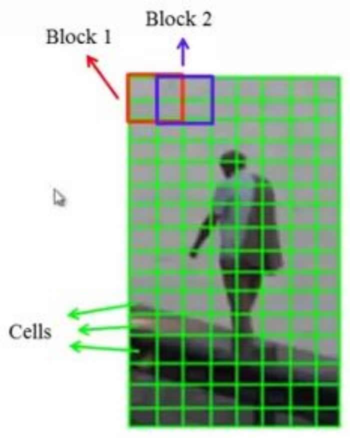


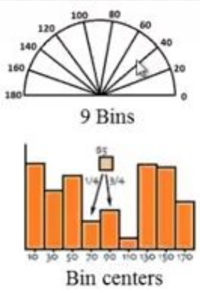




44

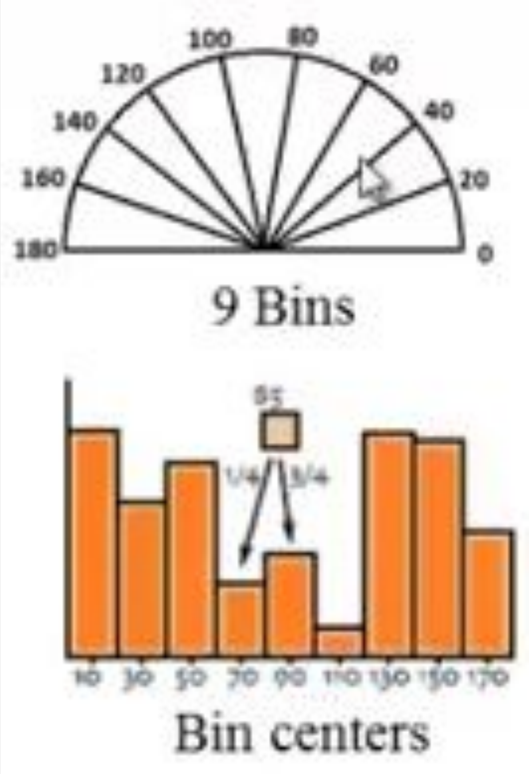
Histogram of Oriented Gradients HOG Feature

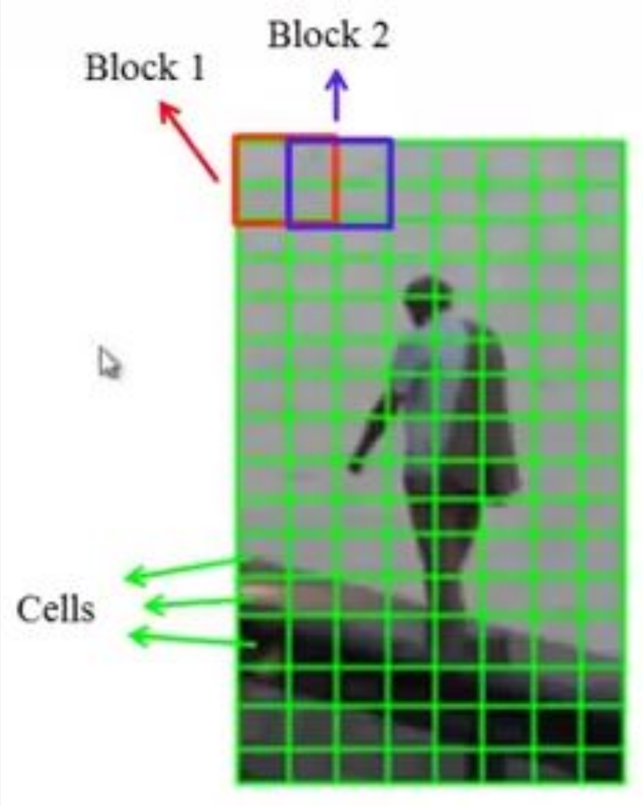


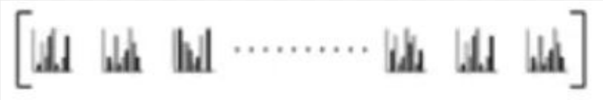


- A global descriptor
- For 64 × 128 image, we divide into 16 × 16 blocks with 50% overlap. That is 7 × 15 blocks = 105 blocks. `7 = (64/16)*2 - 1; 15 = (128/16)*2 - 1.`
- Each block is 2 × 2 cell with 8 × 8 pixels.
- Quantize gradient orientation [0, 180) degrees for each 8x8 to 9 bins (20 degrees each). So, 9 histogram values. **Non-Directional Gradients.**
- Feature for full image is 105 × 4 × 9 = 3,780. Concatenated to a huge feature vector.
- Very popular, used in human detection.

HOG Feature







Gradient Histogram vs. Gray-Scale Histogram?

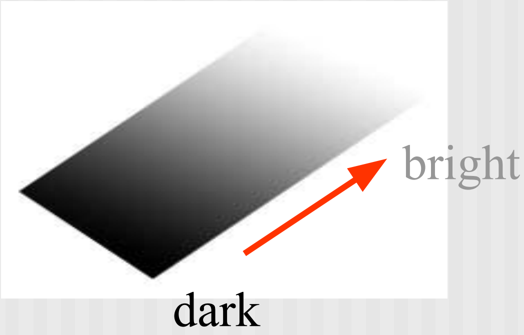
Gradient Magnitude G and Direction θ

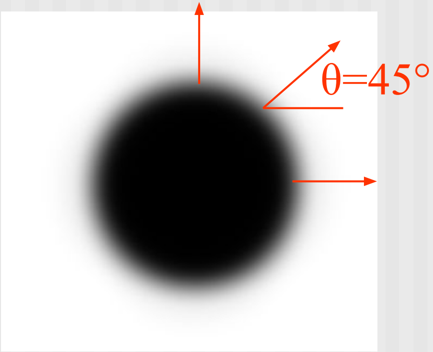
Gradient Direction is perpendicular to edge direction, from dark to bright.

$$\tan\theta = G_y \; / \; G_x$$

$$\theta = \arctan(G_y \; / \; G_x)$$

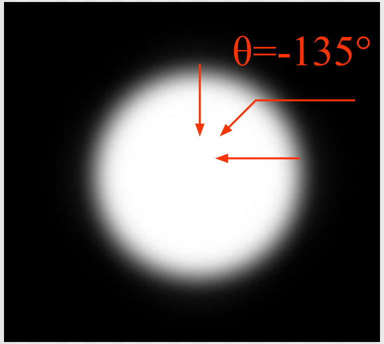
$$G^2 = G_x{}^2 + G_y{}^2$$

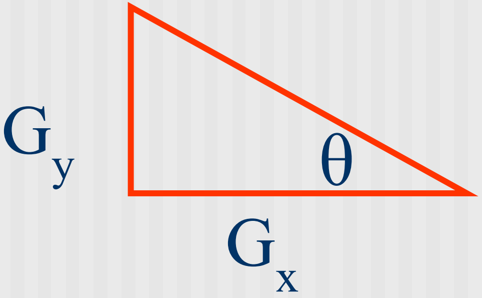




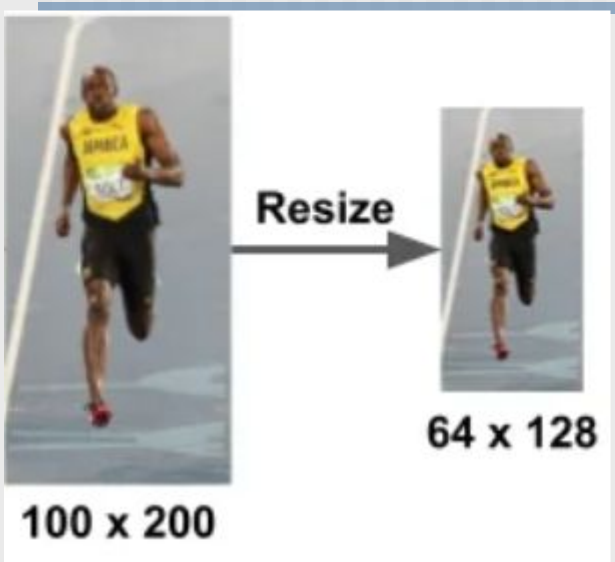
In python **atan(x)** gives angle value between -90° and 90° (add 90 to get 0 to 180).

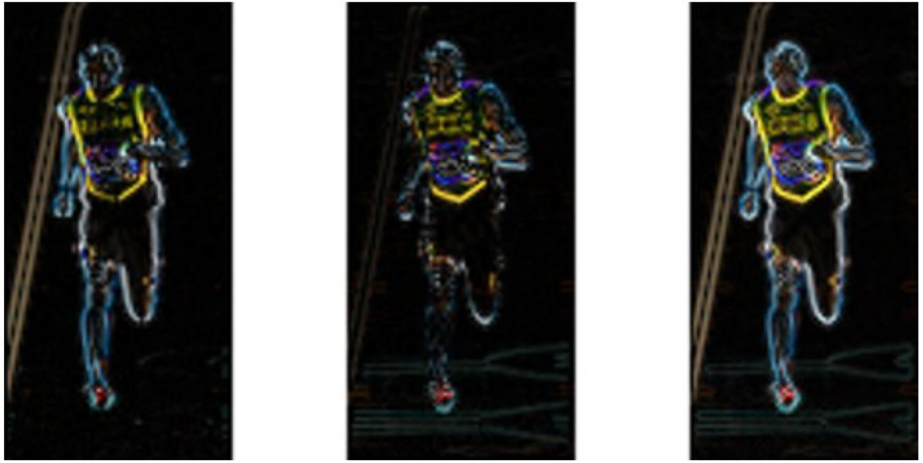
atan2(x) gives angle value between -180° and 180° (add 180 to get 0 to 360).





HOG. Fixed Image Size. Gradient θ and ||G||





Left : Absolute value of x-gradient. Center : Absolute value of y-gradient. Right : Magnitude of gradient.

$$\tan\theta = G_y \ / \ G_x$$

$$\theta = \arctan(G_y \ / \ G_x)$$

$$G^2 = G_x^{\ 2} + G_y^{\ 2}$$

$$||G|| = ||(G_x , G_y)||$$

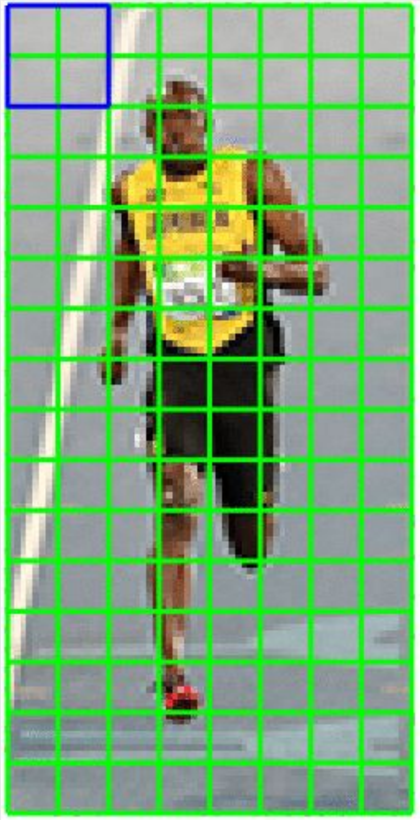
G_x

-1	0	1
-2	0	2
-1	0	1

G_y

1	2	1
0	0	0
-1	-2	-1

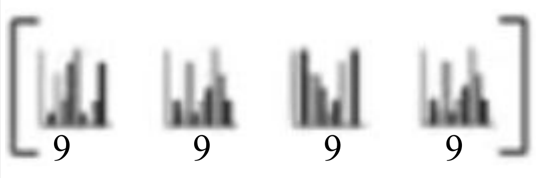
Histogram of Oriented Gradients



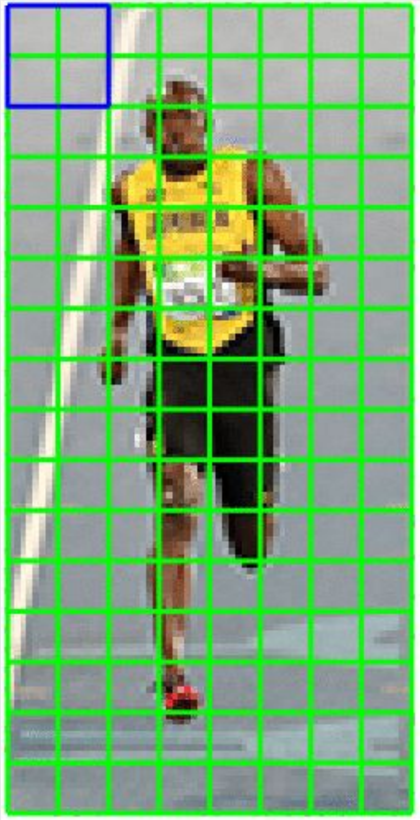
A feature from each blue window, each having 4 cells, **each cell** is a green window with 8x8 pixels.

Each cell outputs 1 histogram which is an array of 9 values. One blue window has **4 cells**.

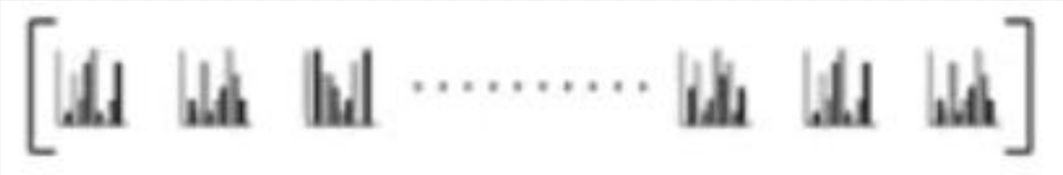
Each blue window feature is, thus, an array of 9 values per cell × 4 cells = 36 values.



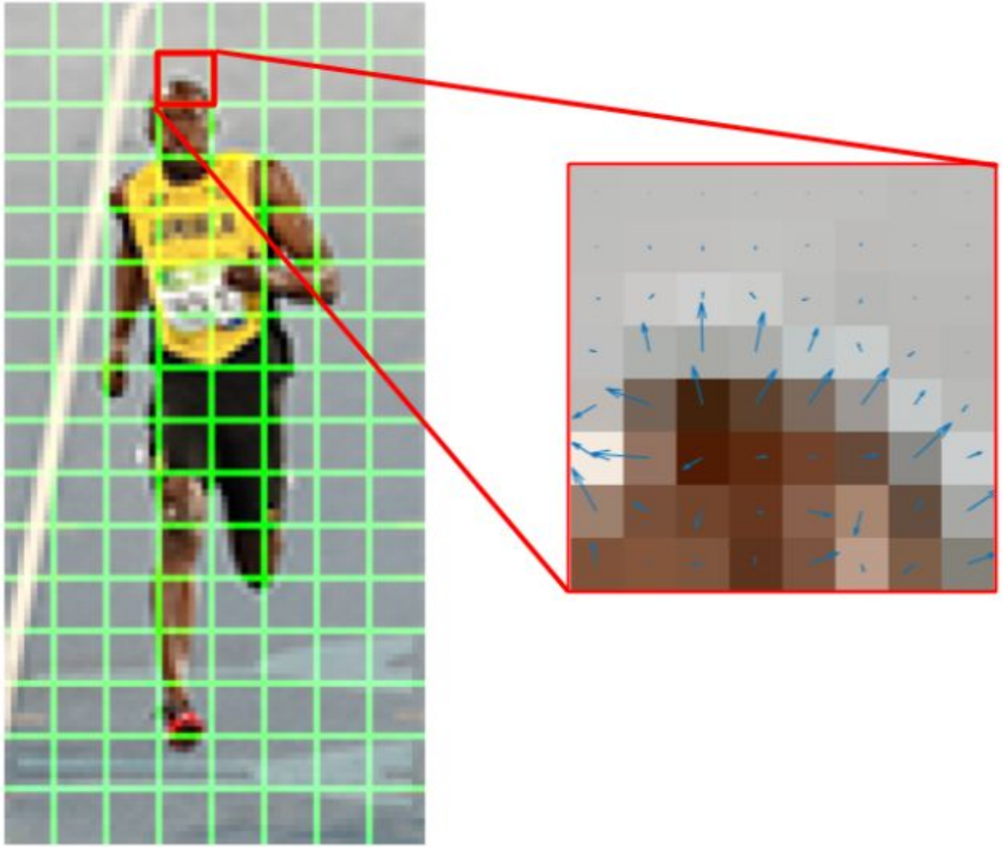
Histogram of Oriented Gradients



- There are 7 horizontal and 15 vertical blue windows, making a total of 7 × 15 = 105 positions or blocks.
- Each 16 × 16 blue block is represented by a 36 × 1 vector (2 × 2 green cells, each of 8 × 8 pixels, each green cell having 9 histograms)
- The HOG feature for this image is a concatenation of 105 such blue box features to get a vector of dimension 3,780 × 1 from 105 × 36.



Histogram of Oriented Gradients



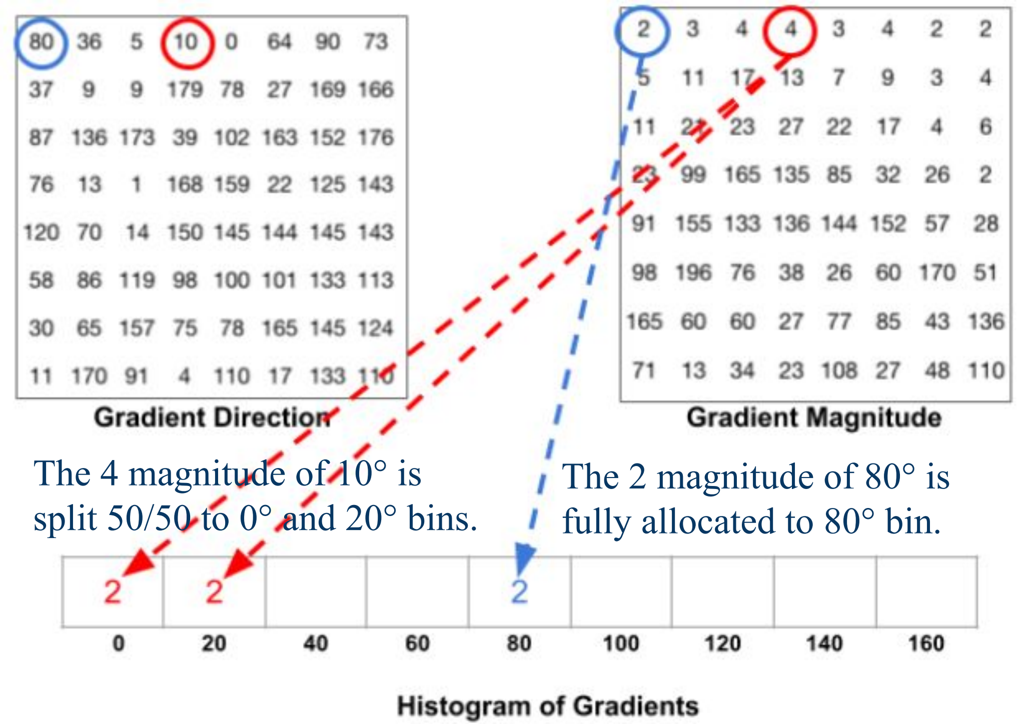
2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	[illegible]	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

Histogram of Oriented Gradients



80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

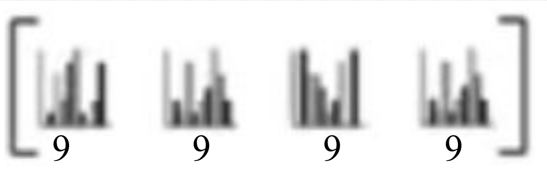
The 4 magnitude of 10° is split 50/50 to 0° and 20° bins.

The 2 magnitude of 80° is fully allocated to 80° bin.

2	2			2				
0	20	40	60	80	100	120	140	160

Histogram of Gradients

Each histogram feature vector from 2×2 concatenation or 36×1 vector is normalized to have a length of 1, but dividing by sqrt of sum squared.



0-180 degree gradients are non-directional gradients. 0 to 360 degrees are directional.

HOG: bin size and what they mean

For each 8x8 region:

- Quantize gradient orientation **[0, 180)** to **9 bins** (20 degrees each). **Non-Directional Gradients.** Use atan(x) + 90 to get [0°, 180°)

0°	20°	40°	60°	80°	100°	120°	140°	160°

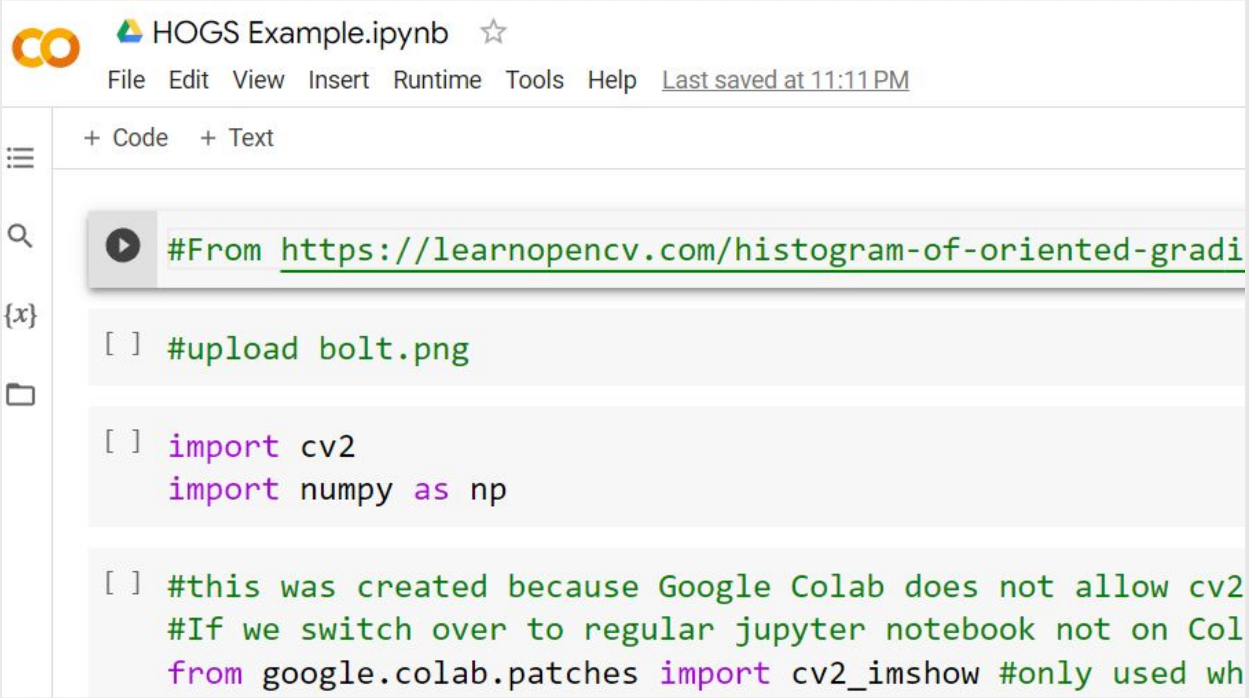
- Quantize gradient orientation **[0, 360)** to **8 bins** (45 degrees each). **Directional Gradients.** Use atan2(x) + 180 to get [0°, 360°).

0°	45°	90°	135°	180°	225°	270°	315°

- Quantize gradient orientation **[0, 360)** to **18 bins** (20 degrees each). **Directional Gradients.** Use atan2(x) + 180 to get [0°, 360°).

0°	20°	40°	60°	80°	100°	120°	140°	160°	180°	200°	220°	240°	260°	280°	300°	320°	340°

HOG Examples



HOGS Example.ipynb
File Edit View Insert Runtime Tools Help Last saved at 11:11 PM
+ Code + Text

```
#From https://learnopencv.com/histogram-of-oriented-gradi
```

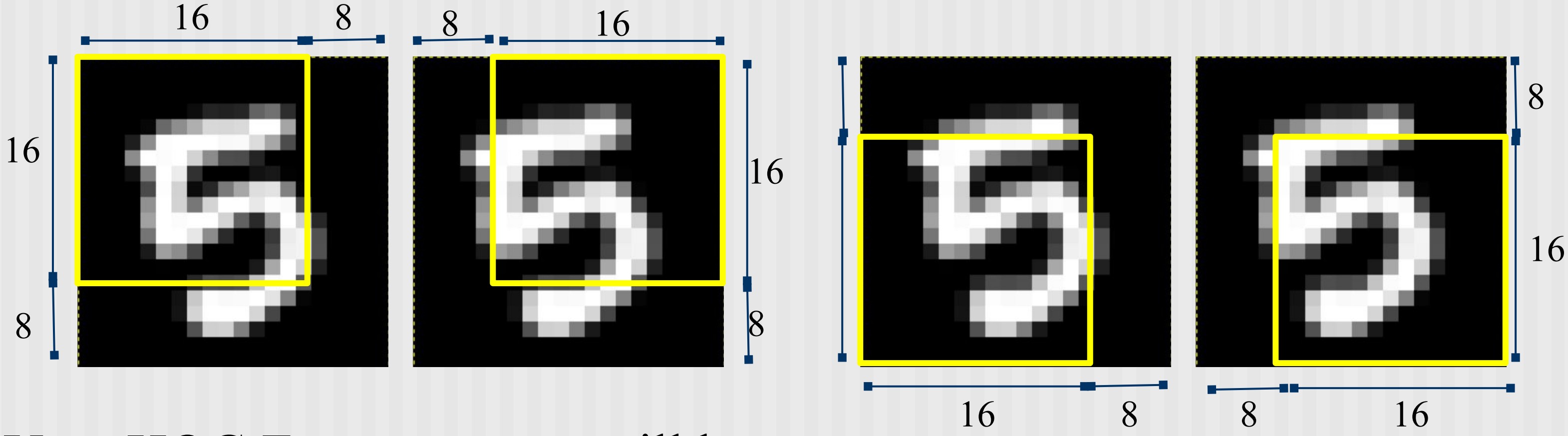
```
#upload bolt.png
```

```
import cv2
import numpy as np
```

```
#this was created because Google Colab does not allow cv2
#If we switch over to regular jupyter notebook not on Col
from google.colab.patches import cv2_imshow #only used wh
```

Quiz 3 to use OpenCV HOG library

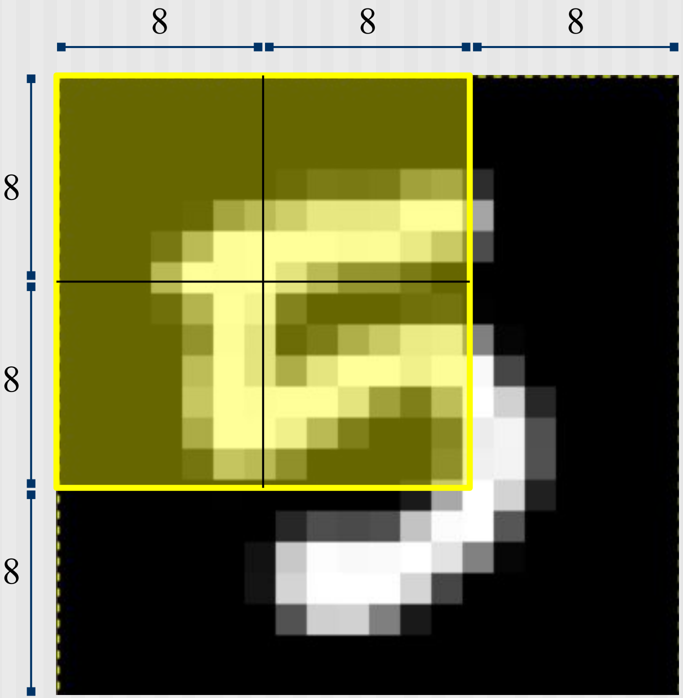
- Scale all 20 × 20 training and testing data into 24 × 24 to start with. Then use 16 × 16 blocks of 4 cells, each 8 × 8. You will have 4 such blocks per image.



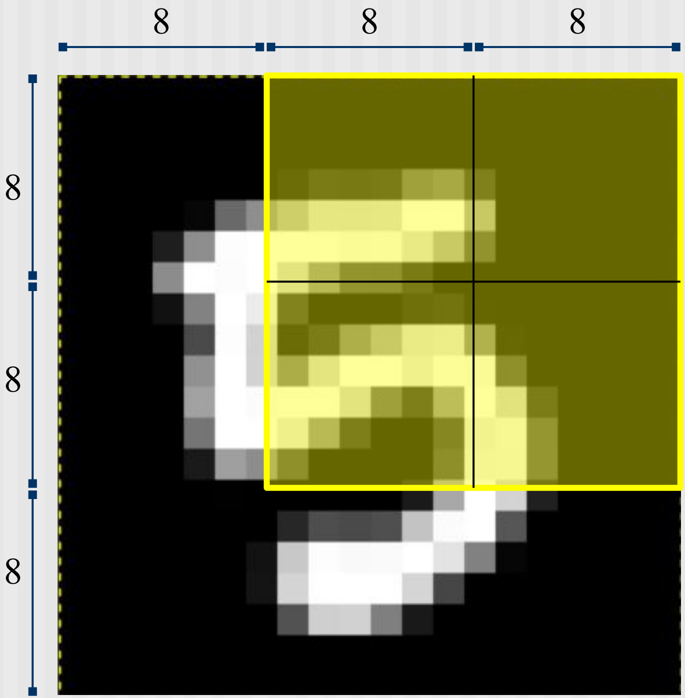
Your HOG Feature vector will have:

- 9 bins per 8x8 cell * 4 cells per block * 4 blocks per image
- Total:144 values.

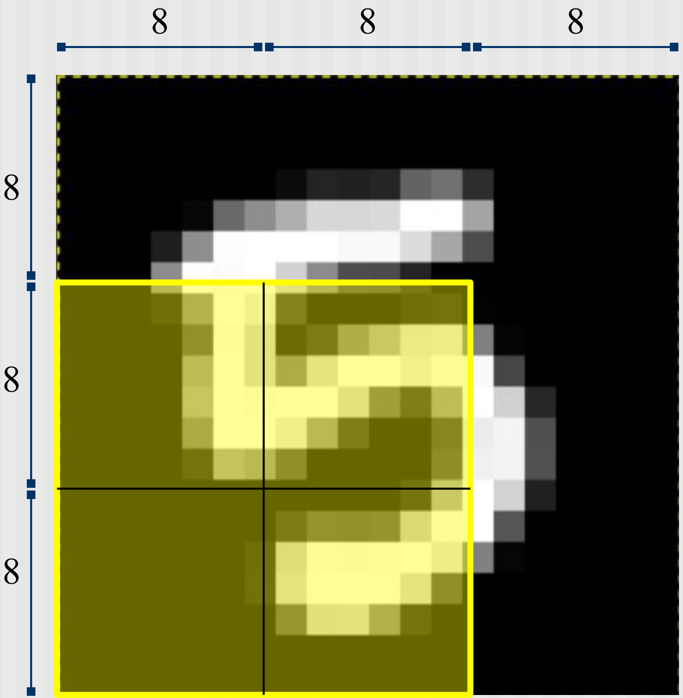
Quiz 3 to use OpenCV HOG library



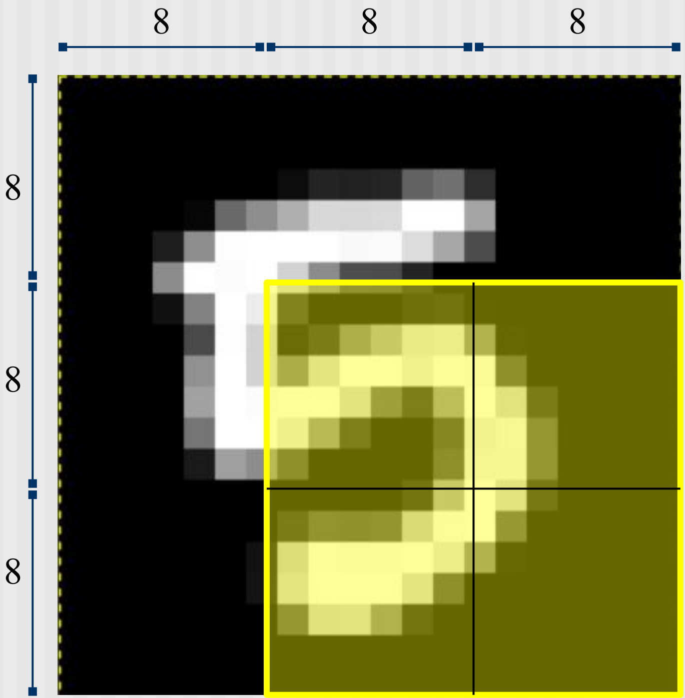
Quiz 3 to use OpenCV HOG library



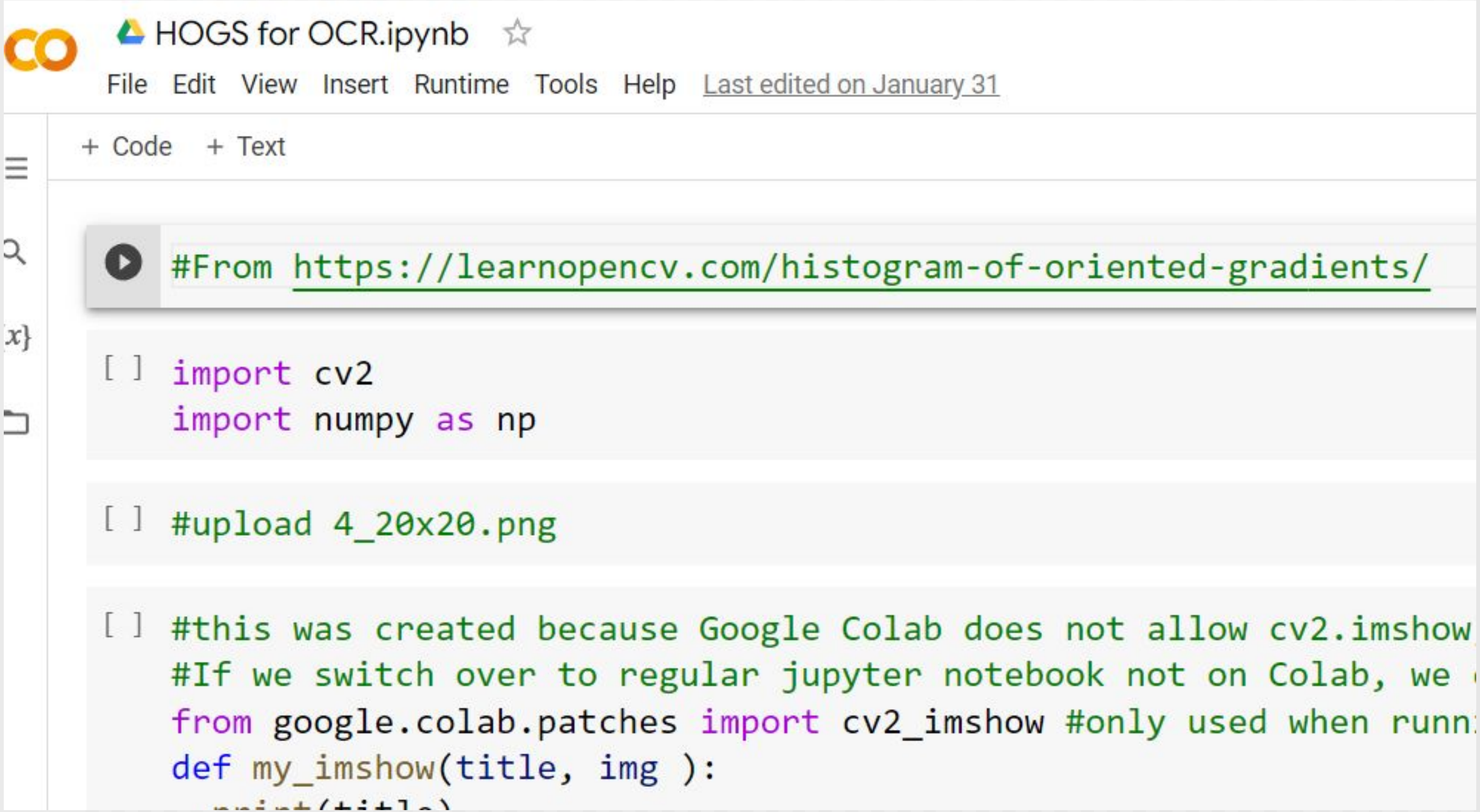
Quiz 3 to use OpenCV HOG library



Quiz 3 to use OpenCV HOG library



Example of 1 Character HOG



```
#From https://learnopencv.com/histogram-of-oriented-gradients/
```

```
import cv2
import numpy as np
```

```
#upload 4_20x20.png
```

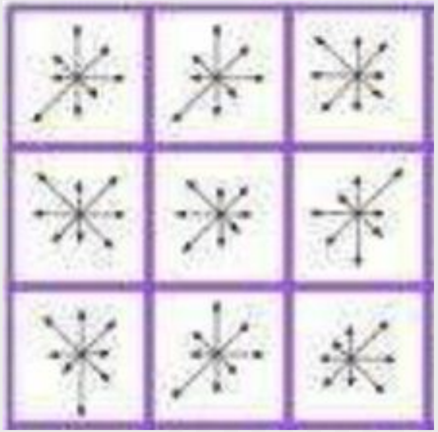
```
#this was created because Google Colab does not allow cv2.imshow
#If we switch over to regular jupyter notebook not on Colab, we 
from google.colab.patches import cv2_imshow #only used when runn
def my_imshow(title, img ):
  print(title)
```

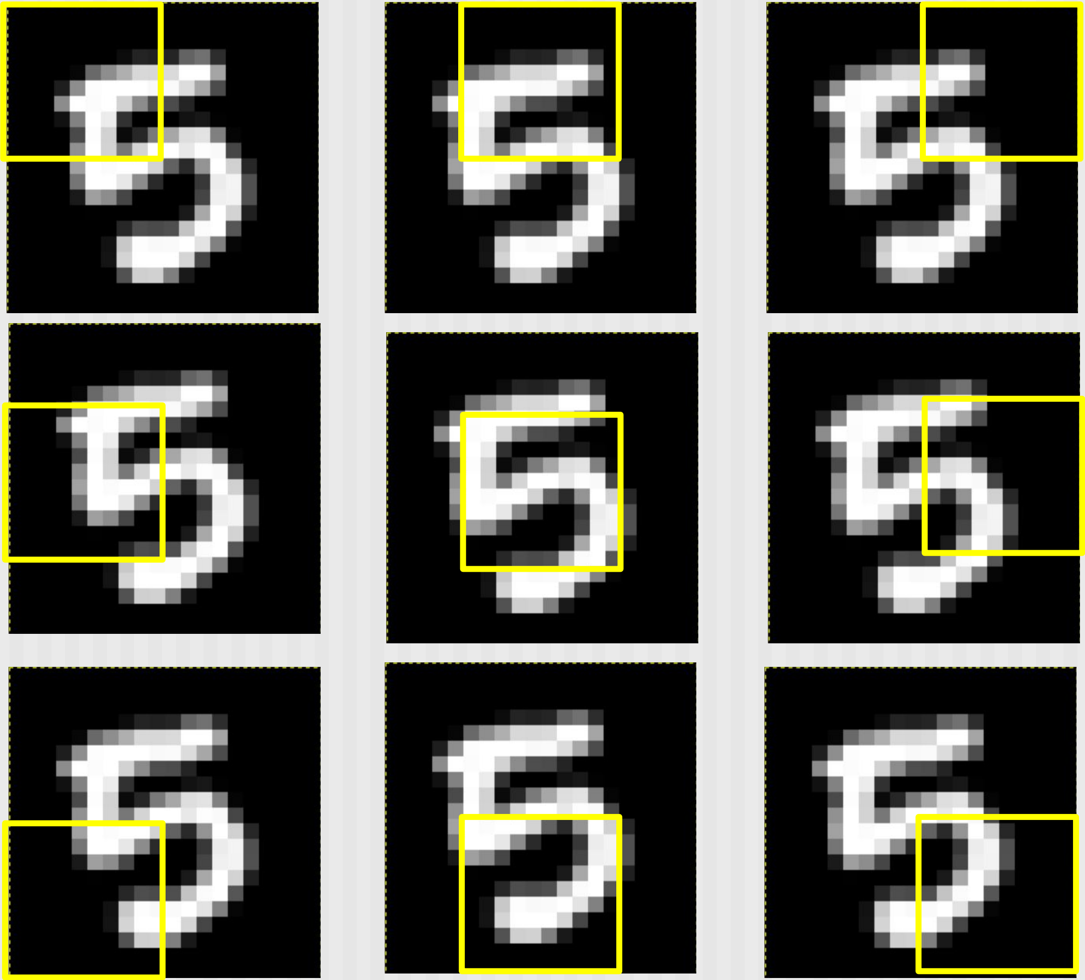
HOG in OpenCV vs. sklearn

- The OpenCV HOG is restricted to 8x8 with 9 bins.
- The sklearn HOG is more generalizable to try 10x10 with 8 bins.

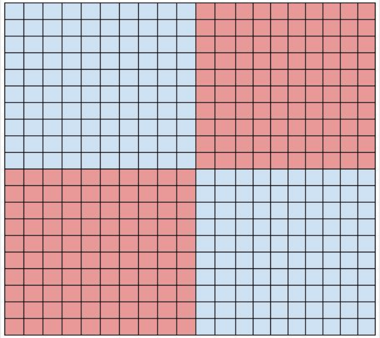
10 x 10 box

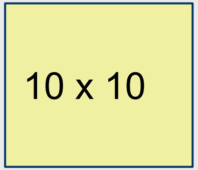
3 x 3 boxes, each 10 x 10 pixels



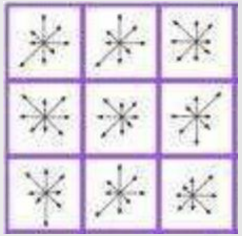


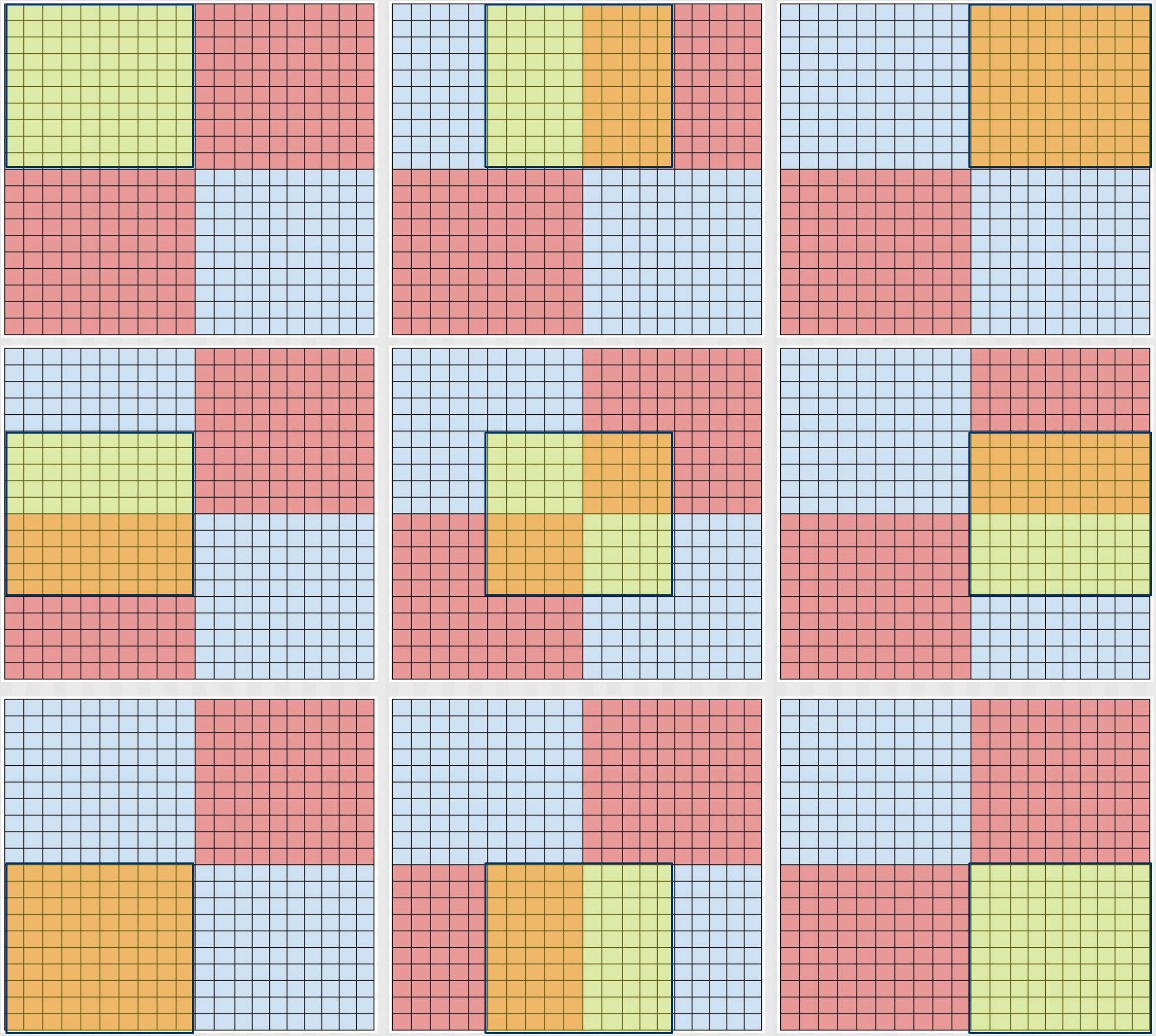
Total: 9 configurations





Output HOG Feature:

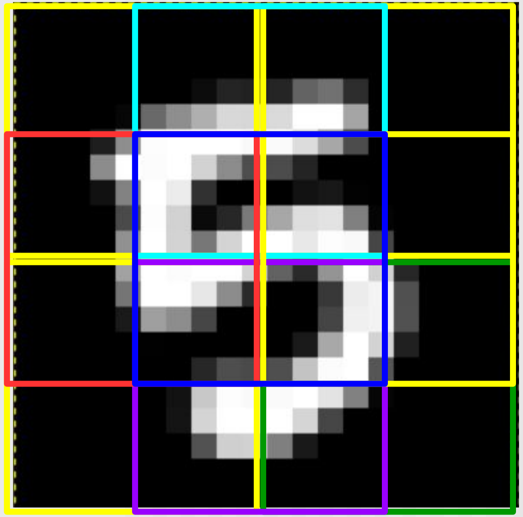




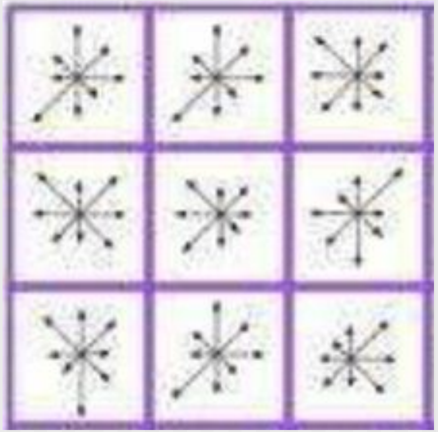
HOG if image were 20 x 20 (use sklearn)

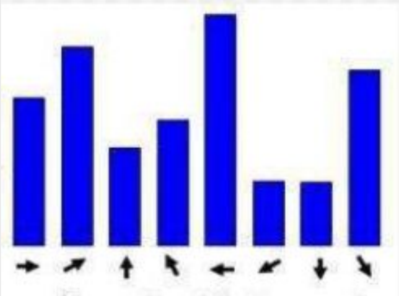
20 x 20 features into 10 x 10 overlapping boxes (shifted by 5 pixels), so 3 x 3 grid. Each grid has 9 bins.

Thus, the 400-pixel color feature vector is now a 3x3x9 = 81 dimensional feature vector of histogram probability.



10 x 10 box





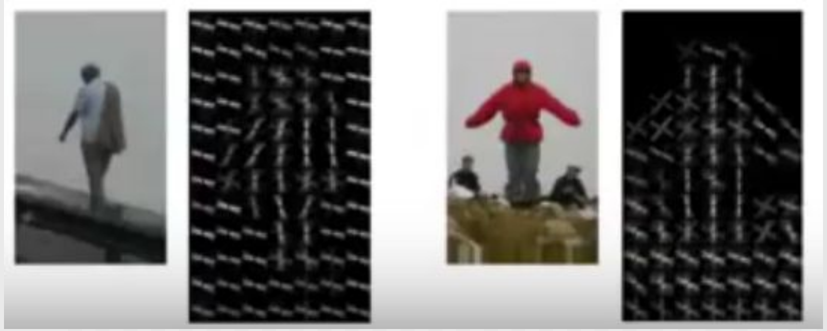
72 = 3 rows x 3 columns x 8 gradient directions

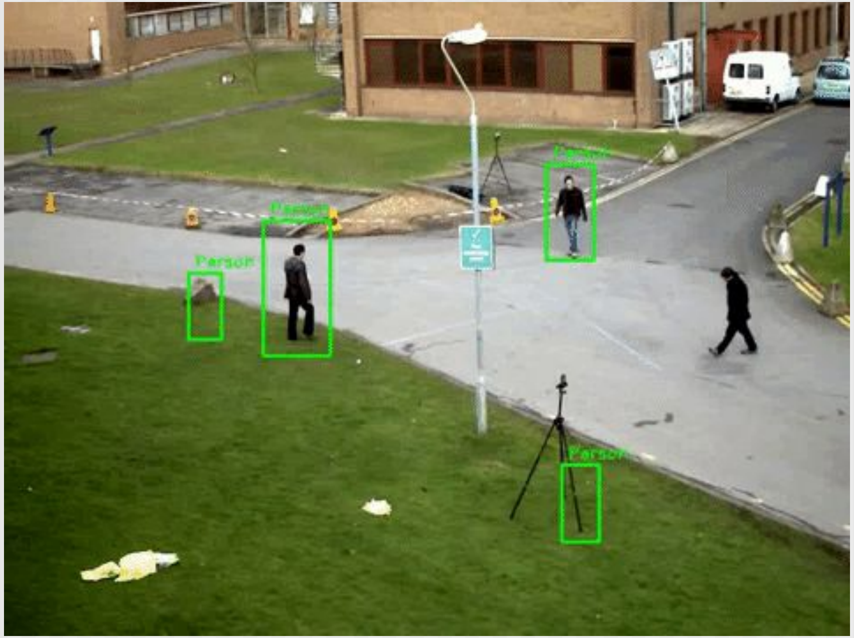
0°	45°	90°	135°	180°	225°	270°	315°

81 = 3 rows x 3 columns x 9 gradient directions

0°	20°	40°	60°	80°	100°	120°	140°	160°

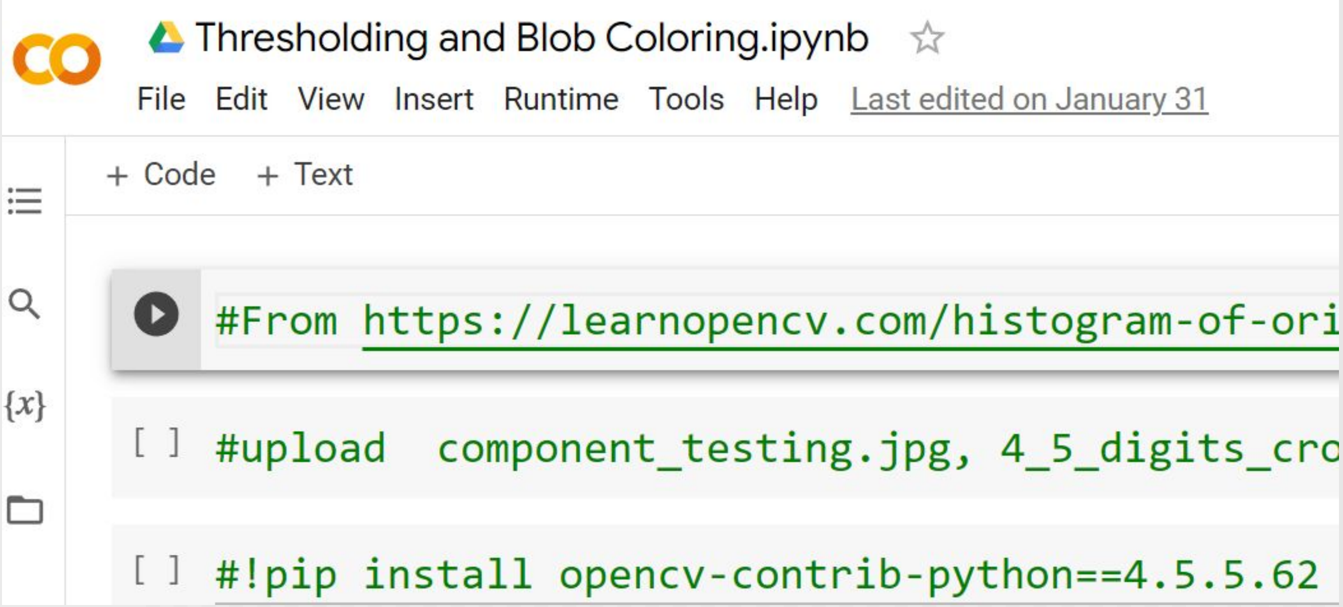
HOG still need to determine how to segment





- Disadvantage of HOG is it covers entire object in the image (global). How to Segment part of the image?

Use Segmentation to Improve OCR



Thresholding and Blob Coloring.ipynb
File Edit View Insert Runtime Tools Help Last edited on January 31
+ Code + Text

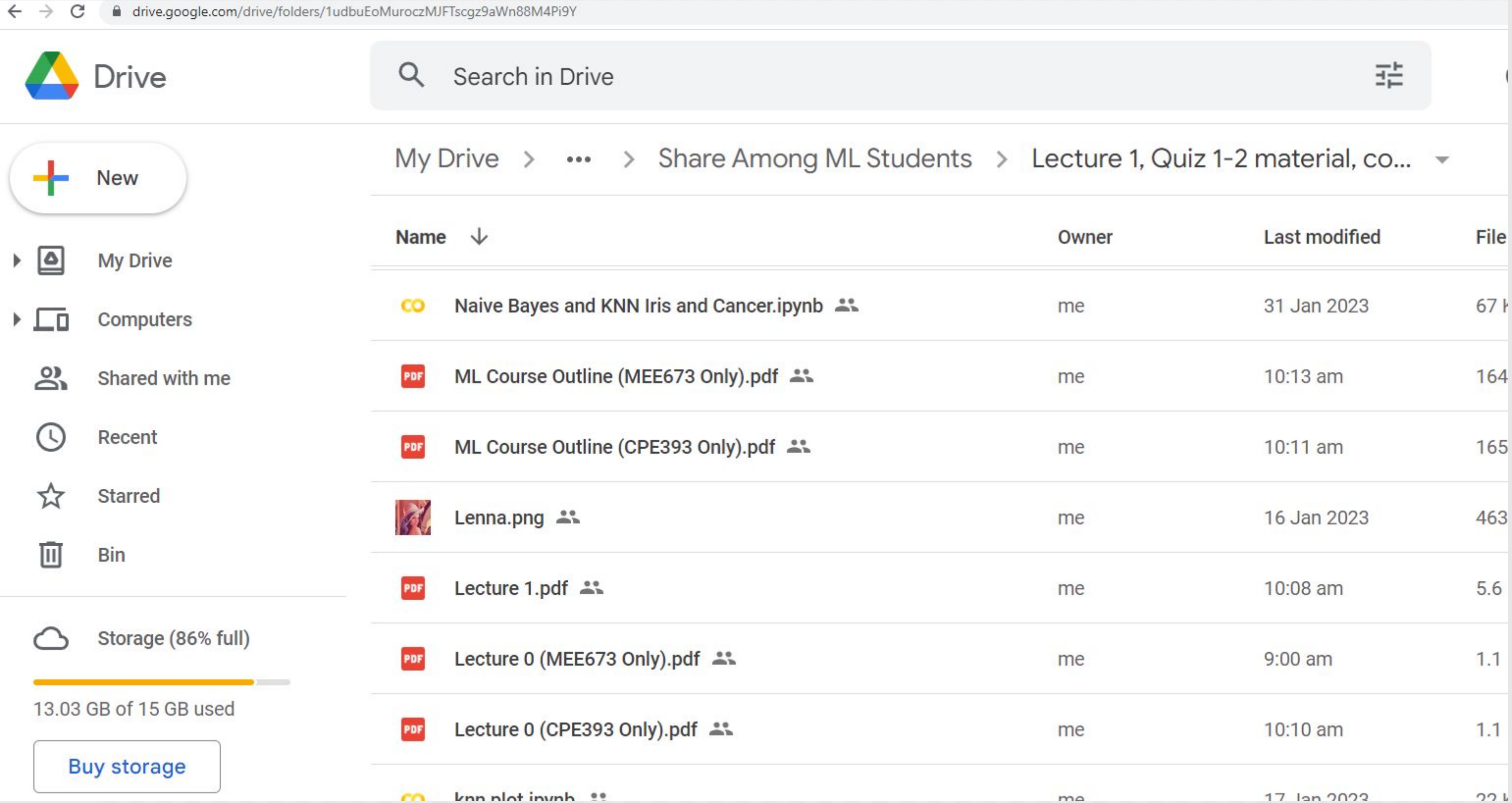
```
#From https://learnopencv.com/histogram-of-ori
```

```
[ ] #upload  component_testing.jpg, 4_5_digits_cro
```

```
[ ] #!pip install opencv-contrib-python==4.5.5.62
```

We can get a bounding box for each training image and scale them to 20 x 20. Do the same for each test image. Then do template matching using gray-scale or using HOG.

New Shared Folder on Google Drive



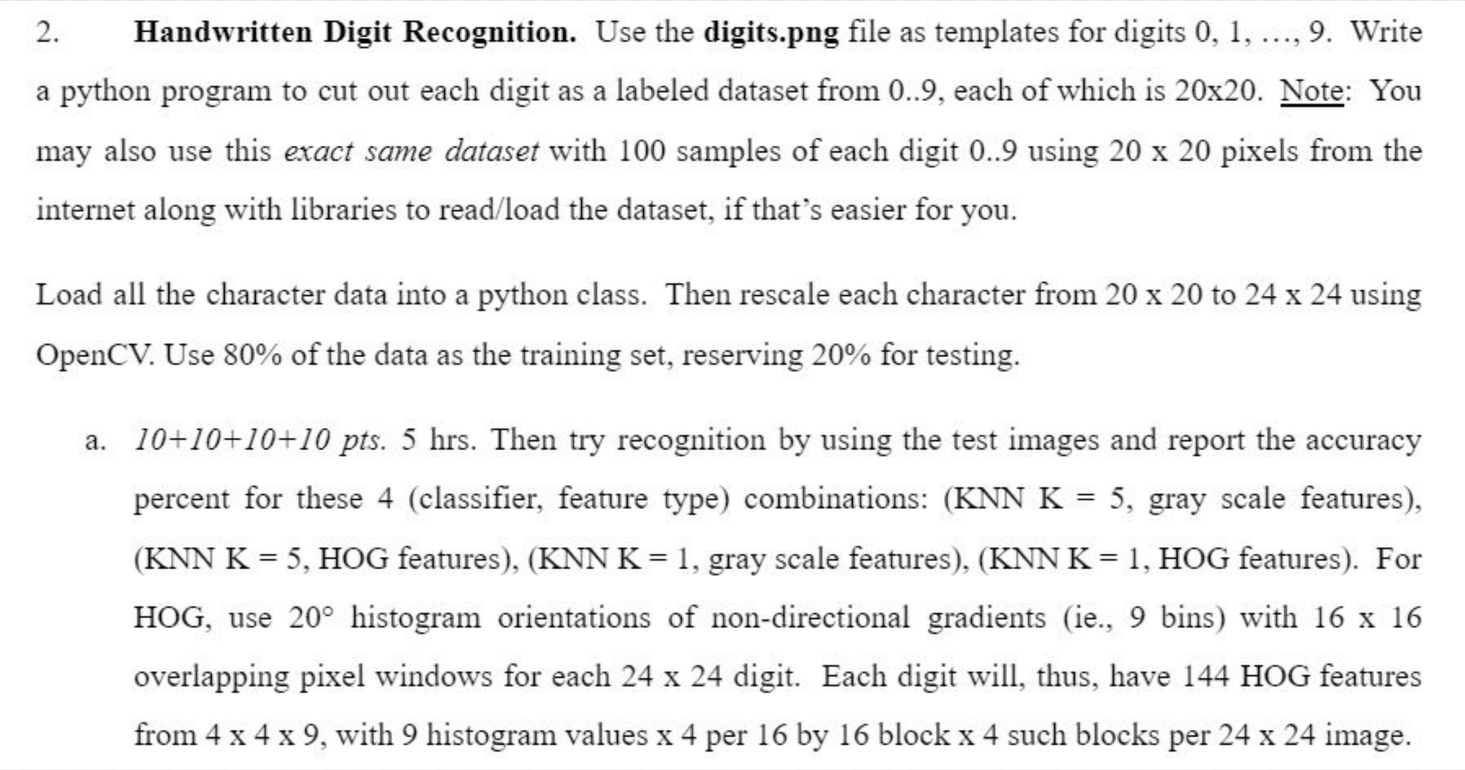
drive.google.com/drive/folders/1udbuEoMuroczMJFTscgz9aWn88M4Pi9Y

My Drive > ••• > Share Among ML Students > Lecture 1, Quiz 1-2 material, co...

13.03 GB of 15 GB used

Name ↓	Owner	Last modified	File
Naive Bayes and KNN Iris and Cancer.ipynb	me	31 Jan 2023	67 k
ML Course Outline (MEE673 Only).pdf	me	10:13 am	164
ML Course Outline (CPE393 Only).pdf	me	10:11 am	165
Lenna.png	me	16 Jan 2023	463
Lecture 1.pdf	me	10:08 am	5.6
Lecture 0 (MEE673 Only).pdf	me	9:00 am	1.1
Lecture 0 (CPE393 Only).pdf	me	10:10 am	1.1
knn plot.ipynb	me	17 Jan 2023	22 k

Quiz 3

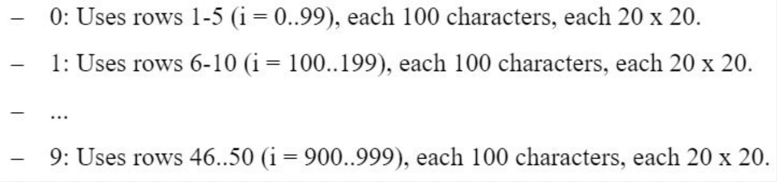


2. **Handwritten Digit Recognition.** Use the **digits.png** file as templates for digits 0, 1, …, 9. Write a python program to cut out each digit as a labeled dataset from 0..9, each of which is 20x20. Note: You may also use this *exact same dataset* with 100 samples of each digit 0..9 using 20 x 20 pixels from the internet along with libraries to read/load the dataset, if that's easier for you.

Load all the character data into a python class. Then rescale each character from 20 x 20 to 24 x 24 using OpenCV. Use 80% of the data as the training set, reserving 20% for testing.

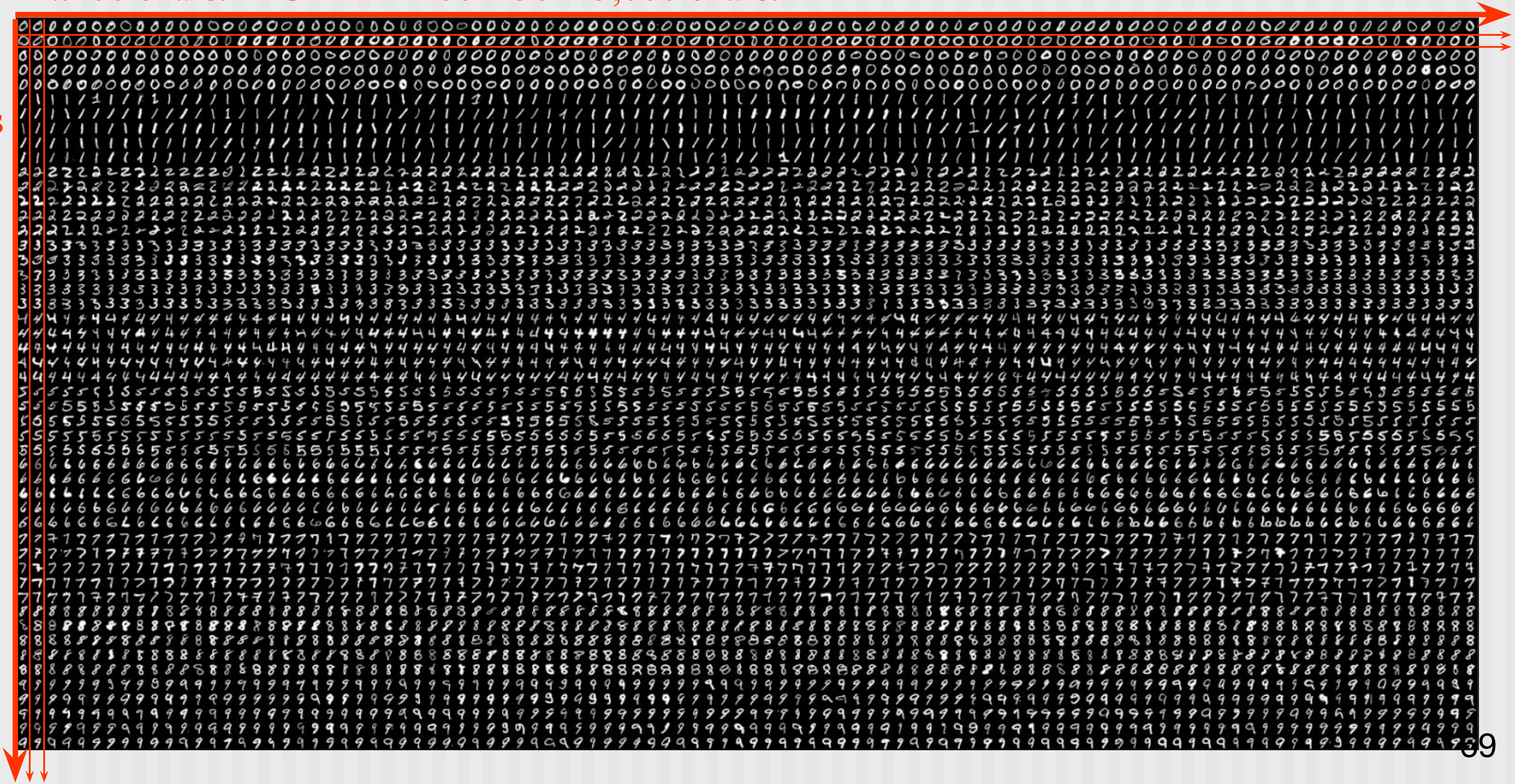
a. *10+10+10+10 pts.* 5 hrs. Then try recognition by using the test images and report the accuracy percent for these 4 (classifier, feature type) combinations: (KNN K = 5, gray scale features), (KNN K = 5, HOG features), (KNN K = 1, gray scale features), (KNN K = 1, HOG features). For HOG, use 20° histogram orientations of non-directional gradients (ie., 9 bins) with 16 x 16 overlapping pixel windows for each 24 x 24 digit. Each digit will, thus, have 144 HOG features from 4 x 4 x 9, with 9 histogram values x 4 per 16 by 16 block x 4 such blocks per 24 x 24 image.

digits.png - 500 samples per digit each is 20x20 pixels



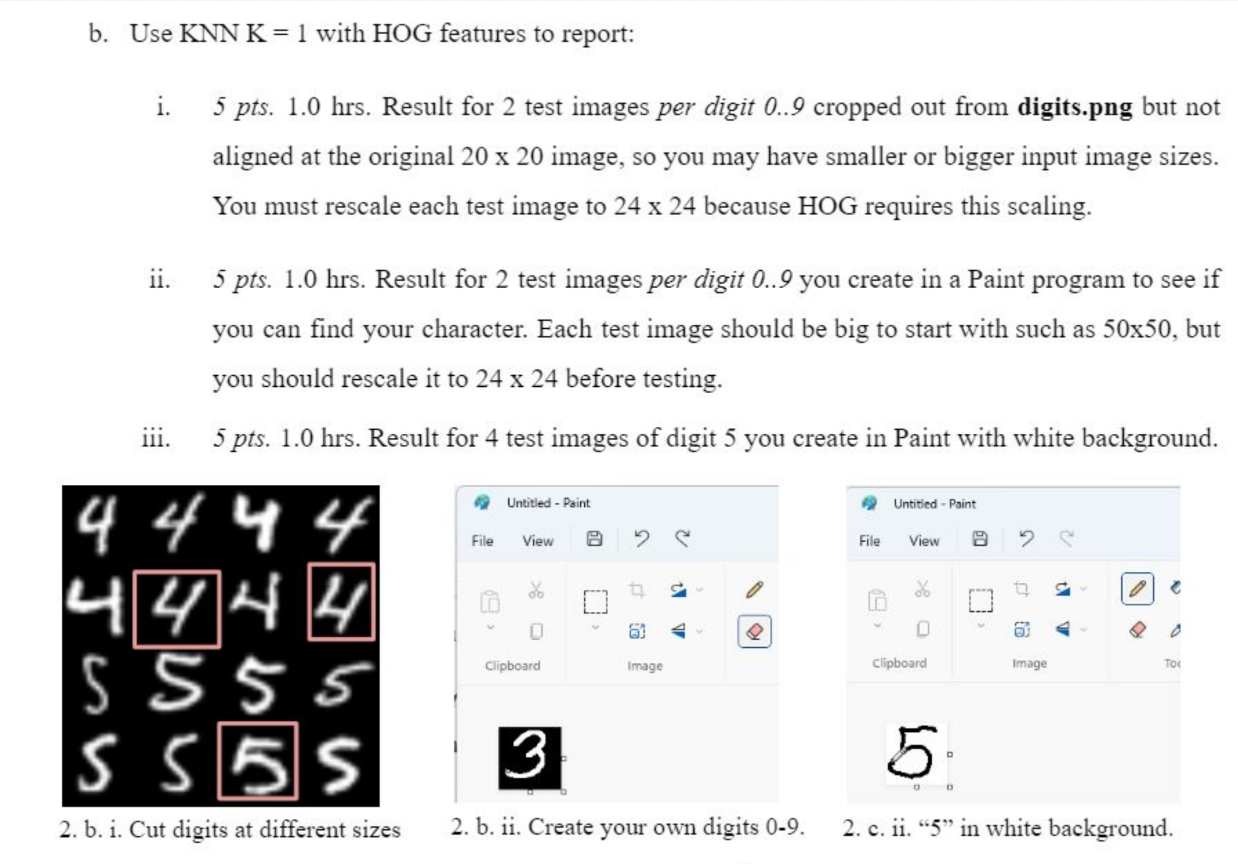
- 0: Uses rows 1-5 (i = 0..99), each 100 characters, each 20 x 20.
- 1: Uses rows 6-10 (i = 100..199), each 100 characters, each 20 x 20.
- ...
- 9: Uses rows 46..50 (i = 900..999), each 100 characters, each 20 x 20.

1..100 chars. TOTAL = 100 x 50 = 5,000 chars.



1..50 chars

Quiz 3



b. Use KNN K = 1 with HOG features to report:

i. *5 pts.* 1.0 hrs. Result for 2 test images *per digit 0..9* cropped out from **digits.png** but not aligned at the original 20 x 20 image, so you may have smaller or bigger input image sizes. You must rescale each test image to 24 x 24 because HOG requires this scaling.

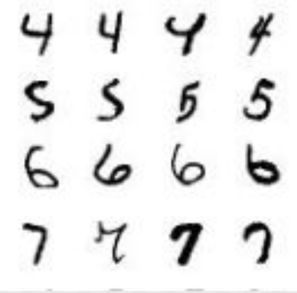
ii. *5 pts.* 1.0 hrs. Result for 2 test images *per digit 0..9* you create in a Paint program to see if you can find your character. Each test image should be big to start with such as 50x50, but you should rescale it to 24 x 24 before testing.

iii. *5 pts.* 1.0 hrs. Result for 4 test images of digit 5 you create in Paint with white background.

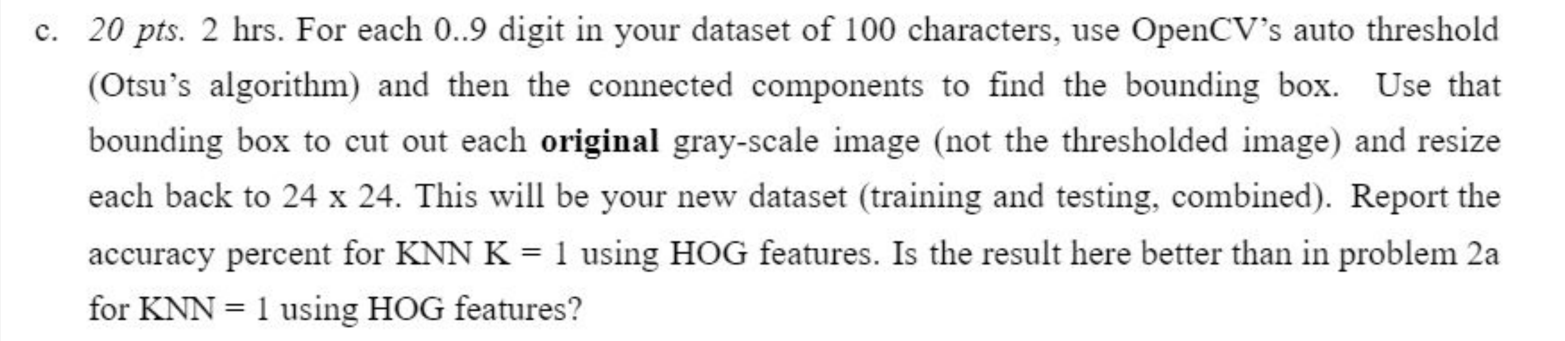
2. b. i. Cut digits at different sizes

2. b. ii. Create your own digits 0-9.

2. c. ii. "5" in white background.



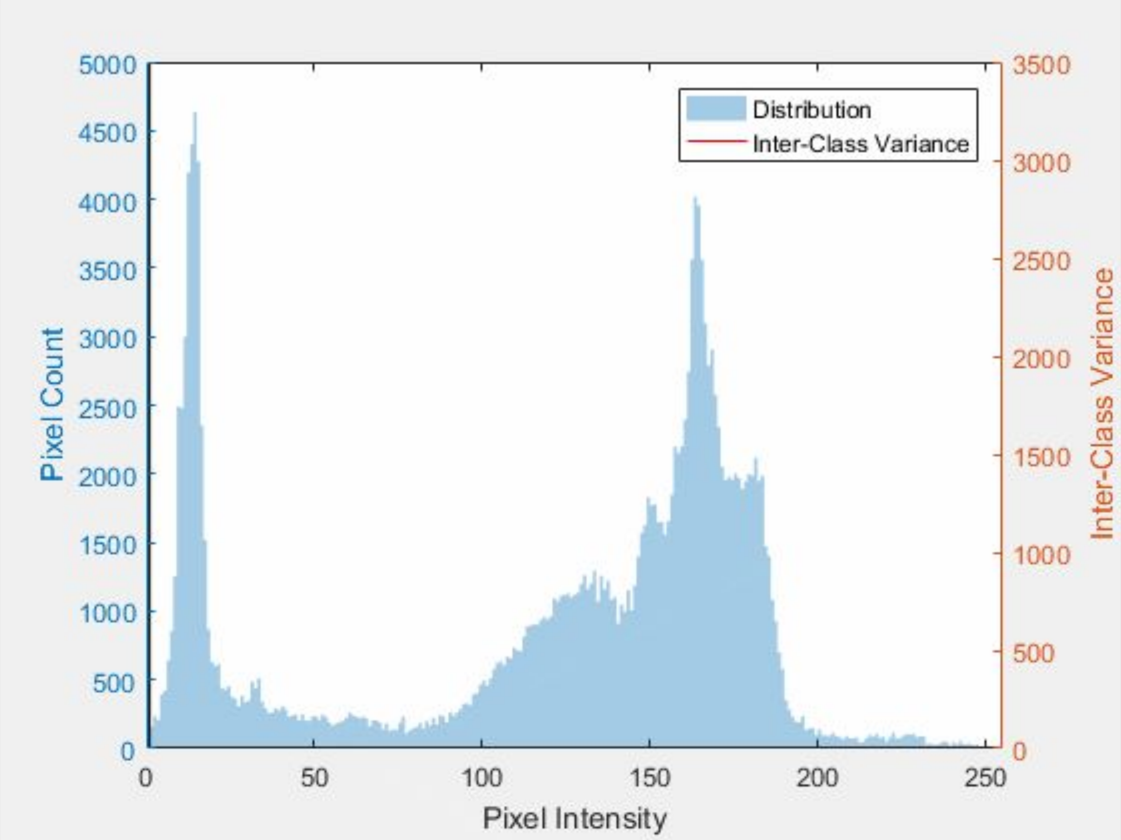
Quiz 3



c. *20 pts.* 2 hrs. For each 0..9 digit in your dataset of 100 characters, use OpenCV's auto threshold (Otsu's algorithm) and then the connected components to find the bounding box. Use that bounding box to cut out each **original** gray-scale image (not the thresholded image) and resize each back to 24 x 24. This will be your new dataset (training and testing, combined). Report the accuracy percent for KNN K = 1 using HOG features. Is the result here better than in problem 2a for KNN = 1 using HOG features?

Otsu’s auto thresholding

https://en.wikipedia.org/wiki/Otsu%27s_method



“Look for the biggest valley”

- area between 2 big mountains.

Uses image histogram: efficient for all image sizes.

Find threshold that will result in the highest inter-class variance.