

ECE 478-578

Intelligent Robotics I

PhD. Husnu Melih Erdogan – Electrical & Computer Engineering

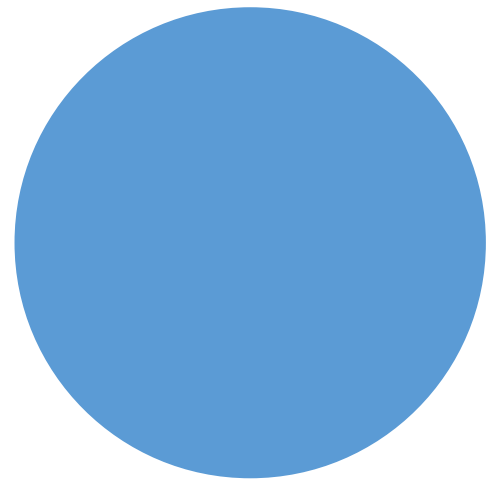
herdogan@pdx.edu

Teaching Assistant



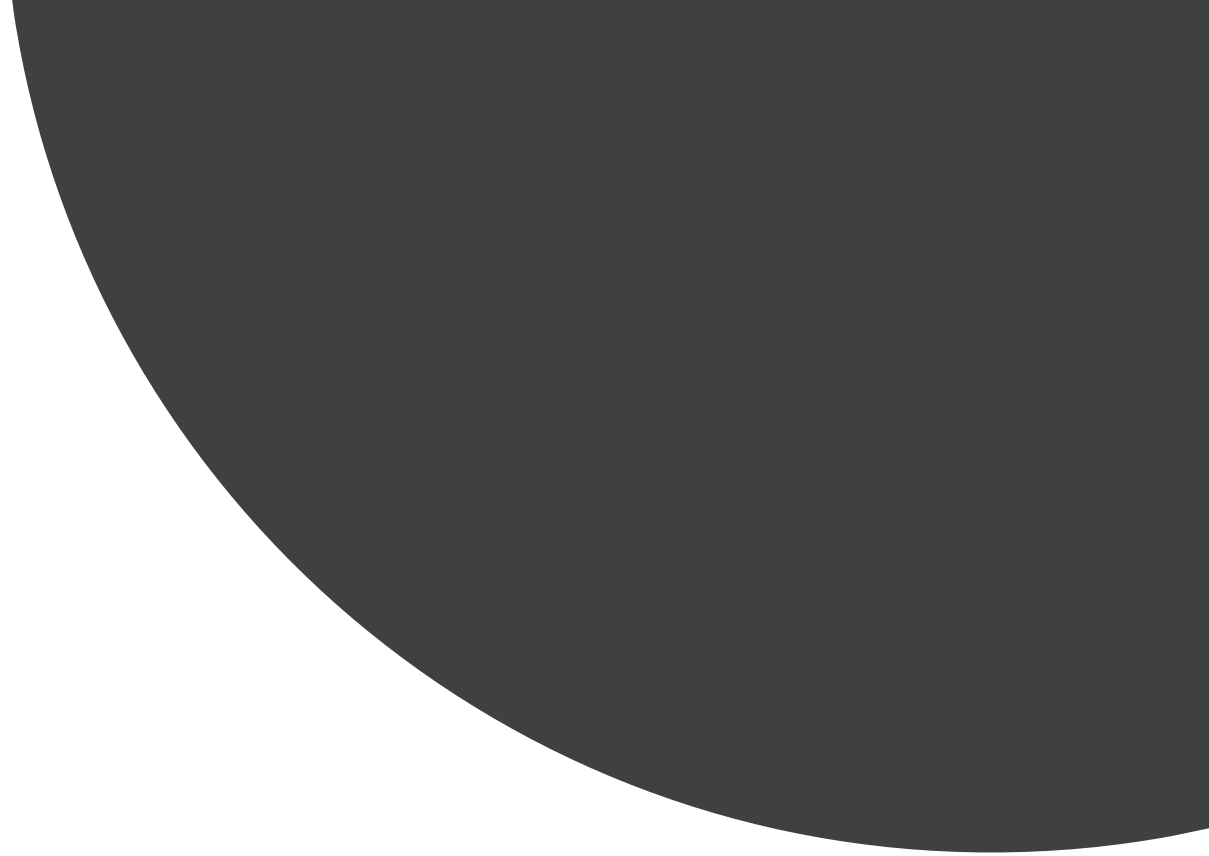
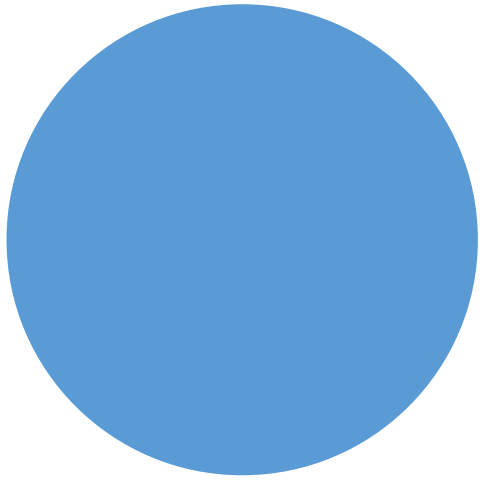
Introduction to OpenCV 3 – Part 3





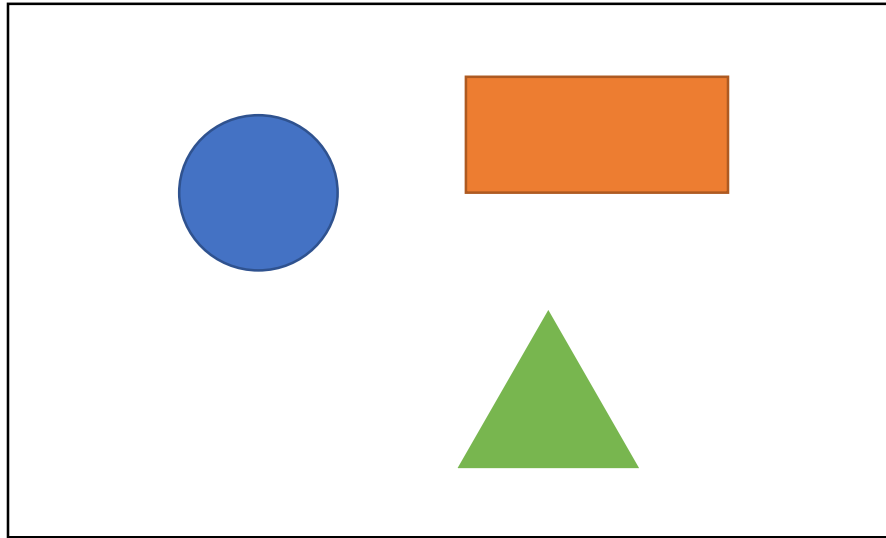
OpenCV - Applications



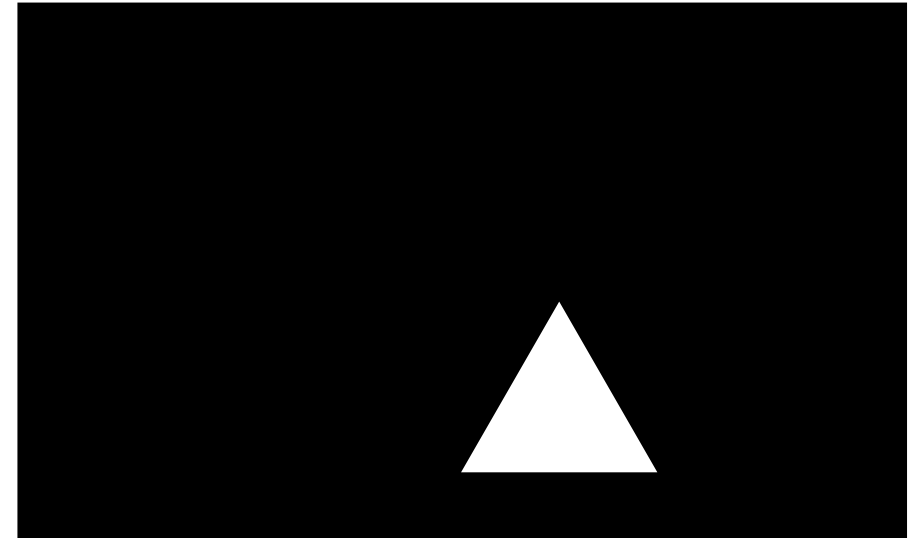
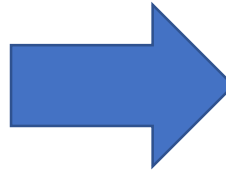


Object Detection by Color

Object Detection by Color

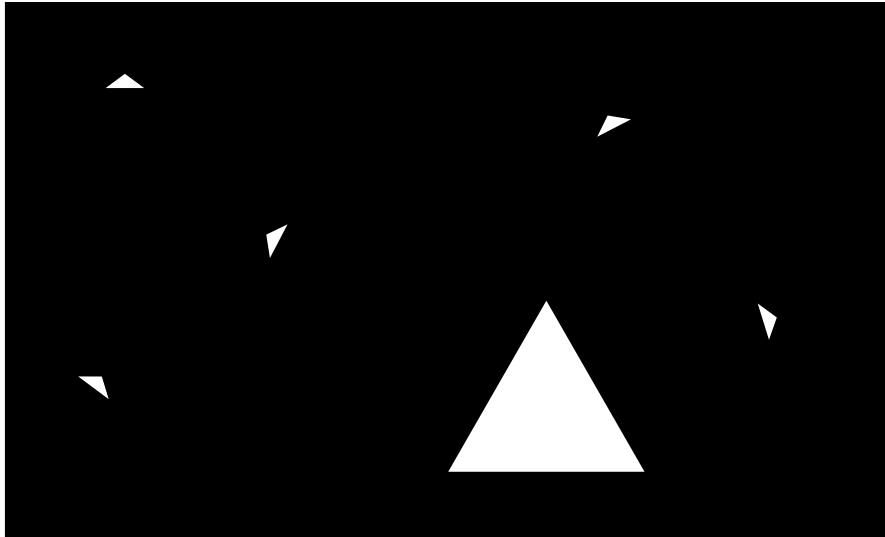


Original RGB Image

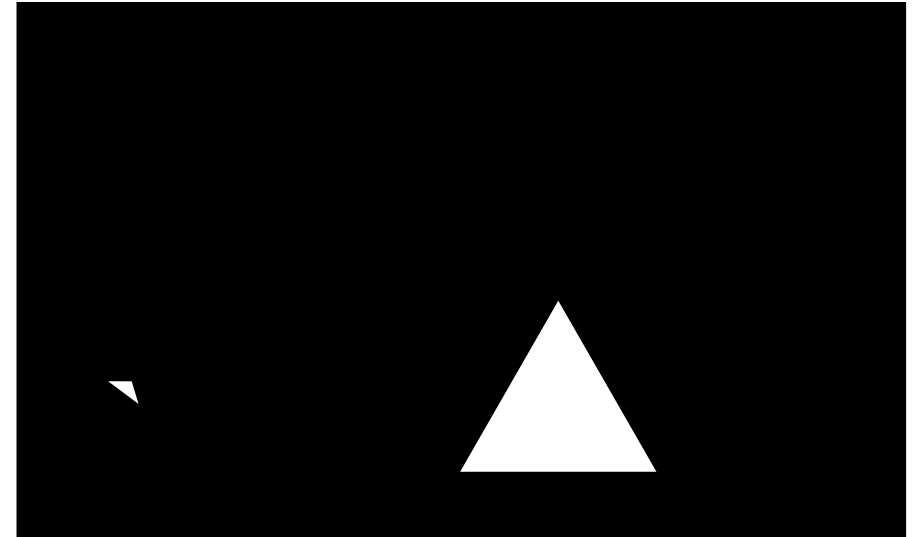
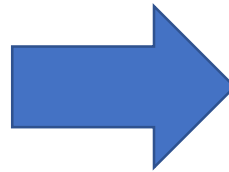


Mask

Object Detection by Color

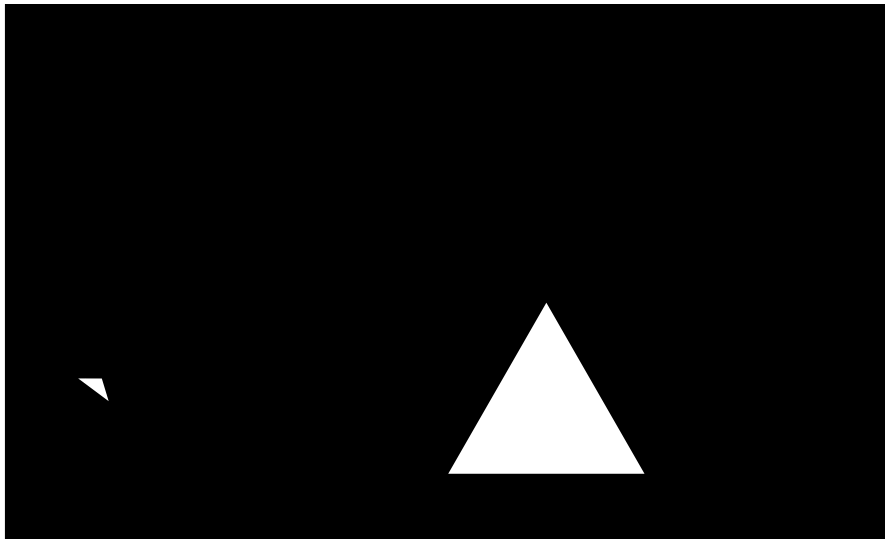


Result of Filtering

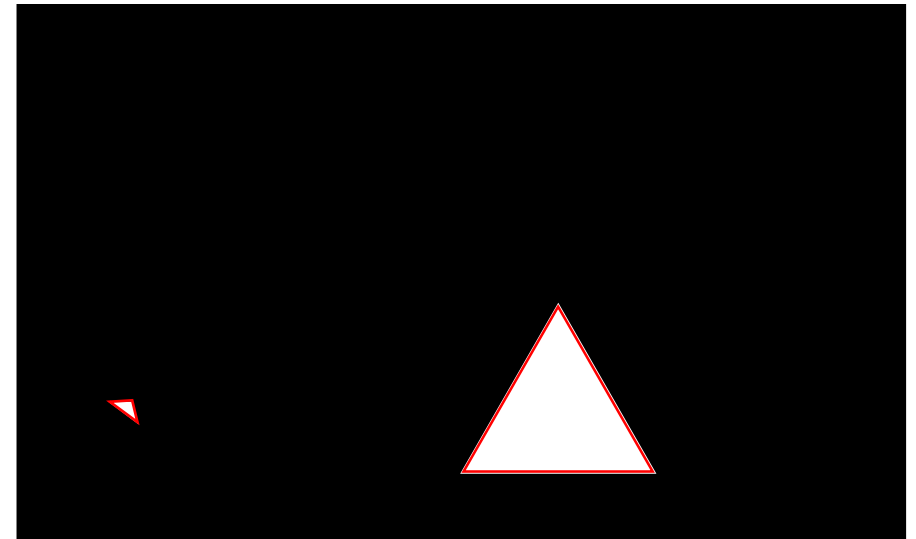
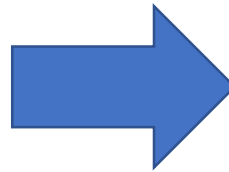


Opening and Closing

Object Detection by Color

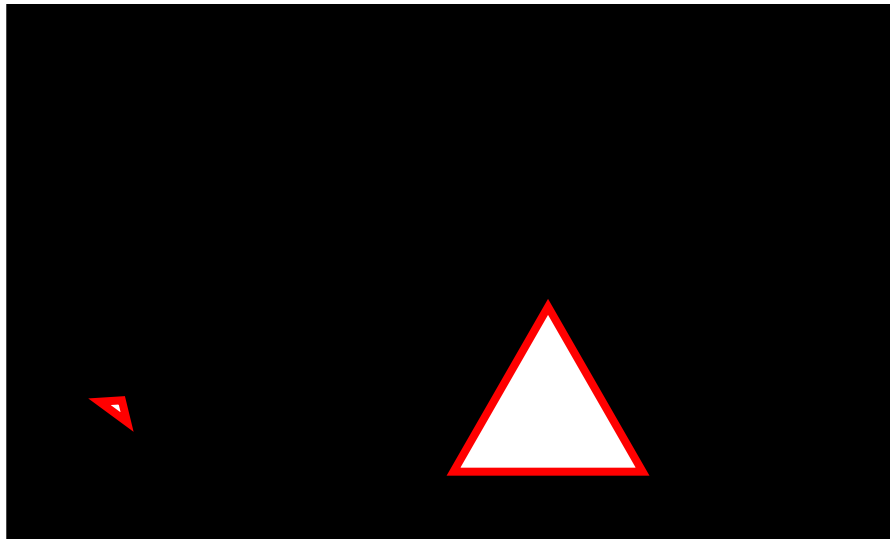


Result of Opening and Closing

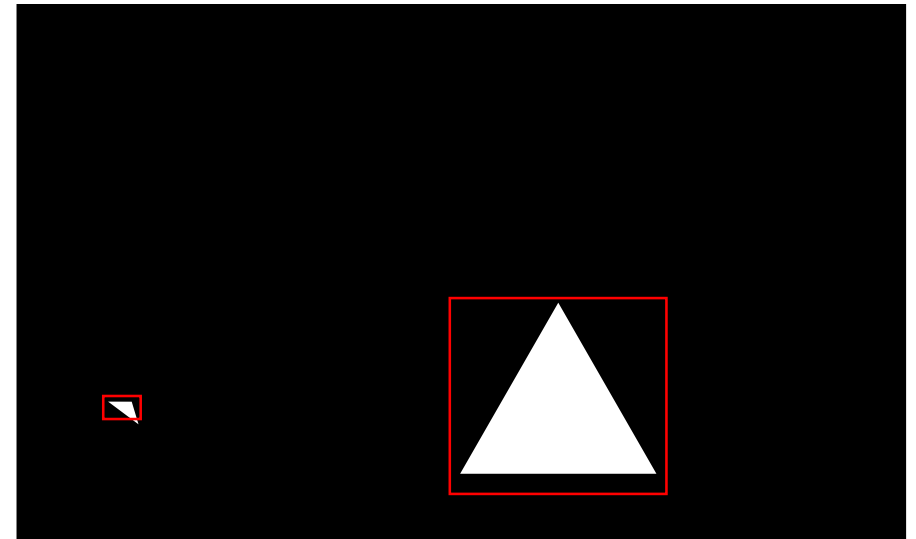
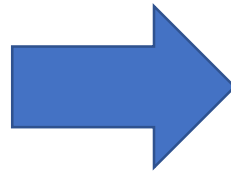


Find Contours

Object Detection by Color



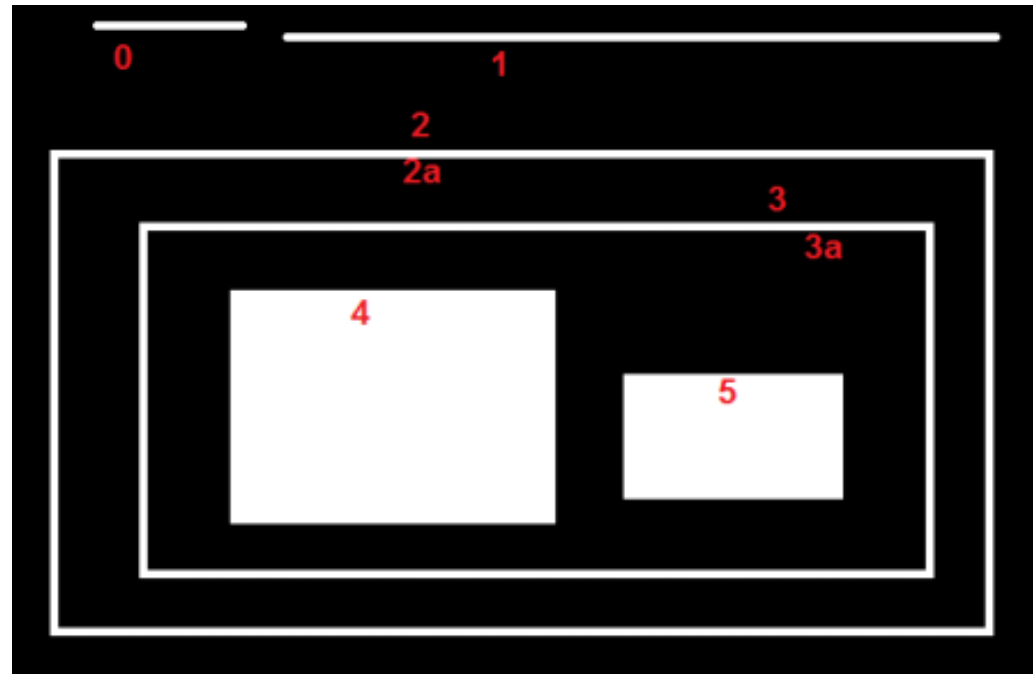
Find Contours



Draw a Bounding Rectangle

Object Detection by Color

- Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition.



Object Detection by Color

- Contour is connected to each other, like, is it child of some other contour, or is it a parent

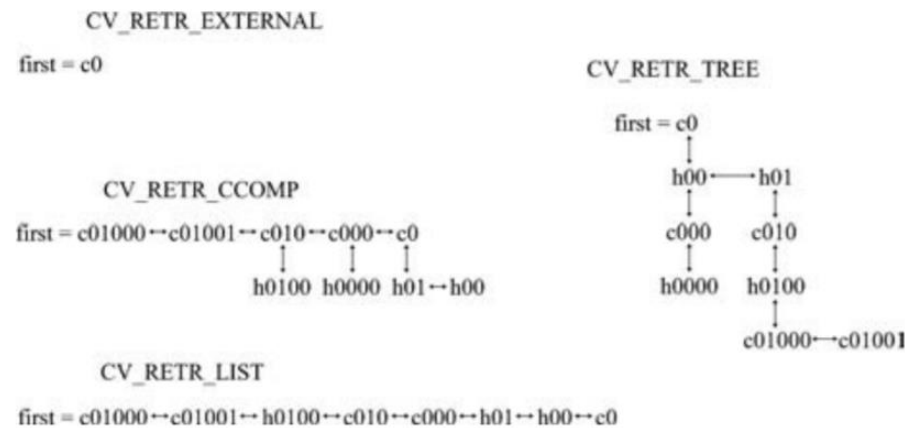


Figure 8-3. The way in which the tree node variables are used to “hook up” all of the contours located by `cvFindContours()`

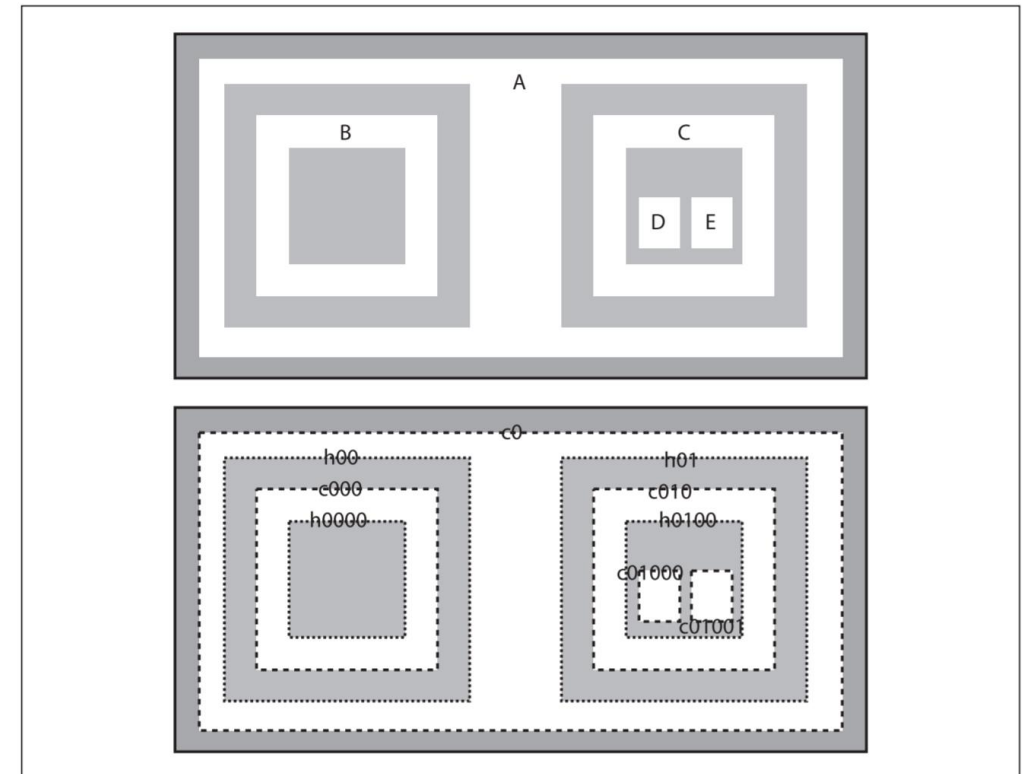
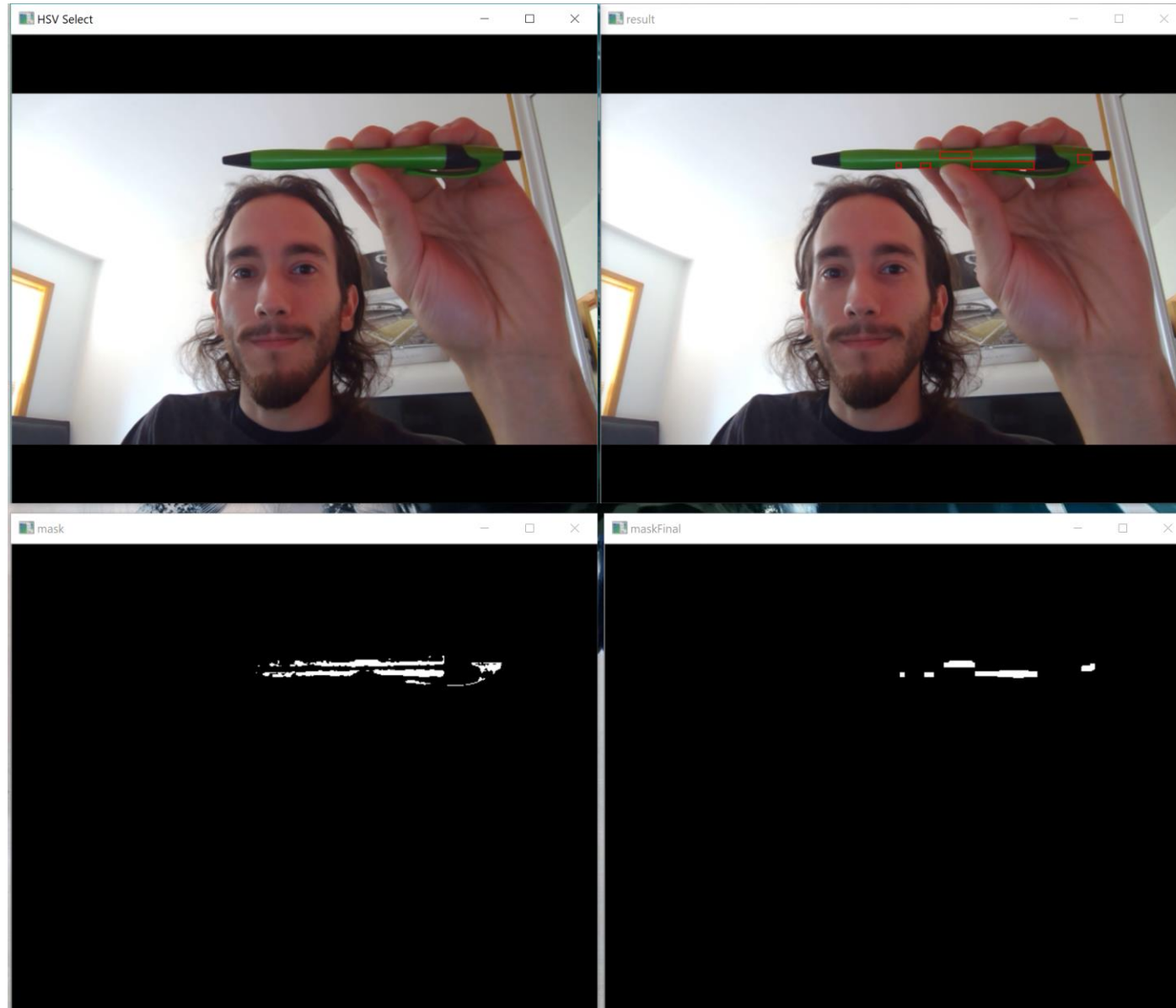


Figure 8-2. A test image (above) passed to `cvFindContours()` (below): the found contours may be either of two types, exterior “contours” (dashed lines) or “holes” (dotted lines)

Object Detection by Color



Object Detection by Color

```
import cv2
import numpy as np

global lowerBound, upperBound
lowerBound=np.array([33,80,40])
upperBound=np.array([102,255,255])

cam = cv2.VideoCapture(0)
kernelOpen=np.ones((5,5))
kernelClose=np.ones((20,20))

def findHSV(event,x,y,flags,param):
    global lowerBound, upperBound
    if event == cv2.EVENT_LBUTTONDOWN:
        ret, img=cam.read()
        img=cv2.resize(img,(600,480))
        rgb = np.uint8([img[y][x]])
        #convert BGR color to HSV values
        HSV= cv2.cvtColor(rgb,cv2.COLOR_BGR2HSV)
        #find lower and upper bounds
        lowerBound=np.array([HSV[0][0][0]-10,HSV[0][0][1]-10,HSV[0][0][2]-40])
        print (lowerBound)
        upperBound=np.array([HSV[0][0][0]+10,HSV[0][0][1]+10,HSV[0][0][2]+40])
        print (upperBound)
```

Object Detection by Color

```
while True:

    ret, img=cam.read()
    img=cv2.resize(img, (600,480))
    cv2.imshow("HSV Select",img)

    #convert BGR to HSV
    imgHSV= cv2.cvtColor(img,cv2.COLOR_BGR2HSV)

    # create the Mask
    mask=cv2.inRange(imgHSV,lowerBound,upperBound)
    cv2.imshow("mask",mask)

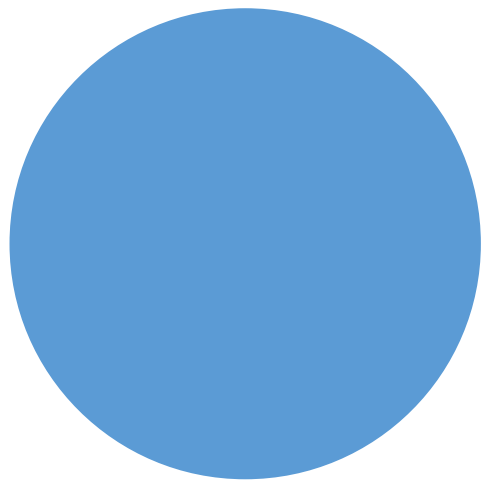
    #morphology to reduce noise
    maskOpen=cv2.morphologyEx(mask,cv2.MORPH_OPEN,kernelOpen)
    maskClose=cv2.morphologyEx(maskOpen,cv2.MORPH_CLOSE,kernelClose)
    maskFinal=maskClose
    cv2.imshow("maskFinal",maskFinal)

    #RETR_EXTERNAL Retrieves only the extreme outer contour
    #CHAIN_APPROX_NONE Translates all the points from the chain code into points.
    img2,conts,h=cv2.findContours(maskFinal.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)

    #draw a rectangle around the object
    for i in range(len(conts)):
        #boundingRect calculates the up-right bounding rectangle of a point set.
        x,y,w,h=cv2.boundingRect(conts[i])
        cv2.rectangle(img, (x,y), (x+w,y+h), (0,0,255), 1)

    #Mouse events
    cv2.namedWindow('HSV Select')
    cv2.setMouseCallback('HSV Select',findHSV)

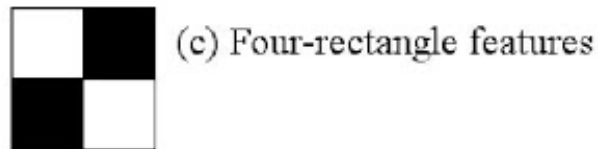
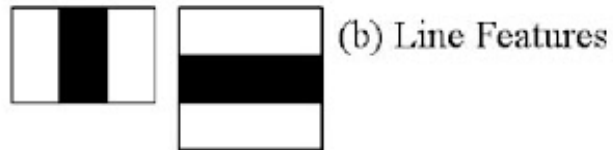
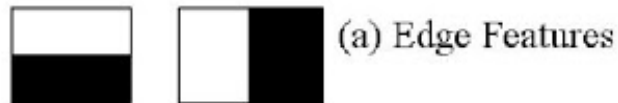
    cv2.imshow("result",img)
    cv2.waitKey(1)
    k = cv2.waitKey(1) & 0xFF
    if k == 27:
        break
```



Face Detection

Viola Jones Face Detection

- Looking for most relevant features
 - Eyes are darker than forehead
 - Nose is darker than cheeks etc.
- Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.
- They are called Haar Features



Viola Jones Face Detection

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

ideal **Haar-feature**
pixel intensities

0.1	0.2	0.6	0.8
0.2	0.3	0.8	0.6
0.2	0.1	0.6	0.8
0.2	0.1	0.8	0.9

these are real values
detected on an image

$$\Delta = \text{dark} - \text{white} = \frac{1}{n} \sum_{\text{dark}}^n I(x) - \frac{1}{n} \sum_{\text{white}}^n I(x)$$

Δ for ideal Haar-feature is **1**

Δ for the real image: **0.74** – **0.18** = **0.56**

Viola Jones Face Detection – Integral Image

- All possible sizes and locations of each kernel is used
- This is very slow, so they used Integral Image to speed the algorithm up

Original Image

3	8	2	1
6	3	9	7
5	2	4	9
6	0	1	8

Integral Image

Viola Jones Face Detection – Integral Image

- Only bottom right value is used
- It is more optimal

Original Image

3	8	2	1
6	3	9	7
5	2	4	9
6	0	1	8

Integral Image

3	11	13	14
9	20	31	39
14	27	42	59
20	33	49	74

Viola Jones Face Detection

Original Image

3	8	2	1
6	3	9	7
5	2	4	9
6	0	1	8

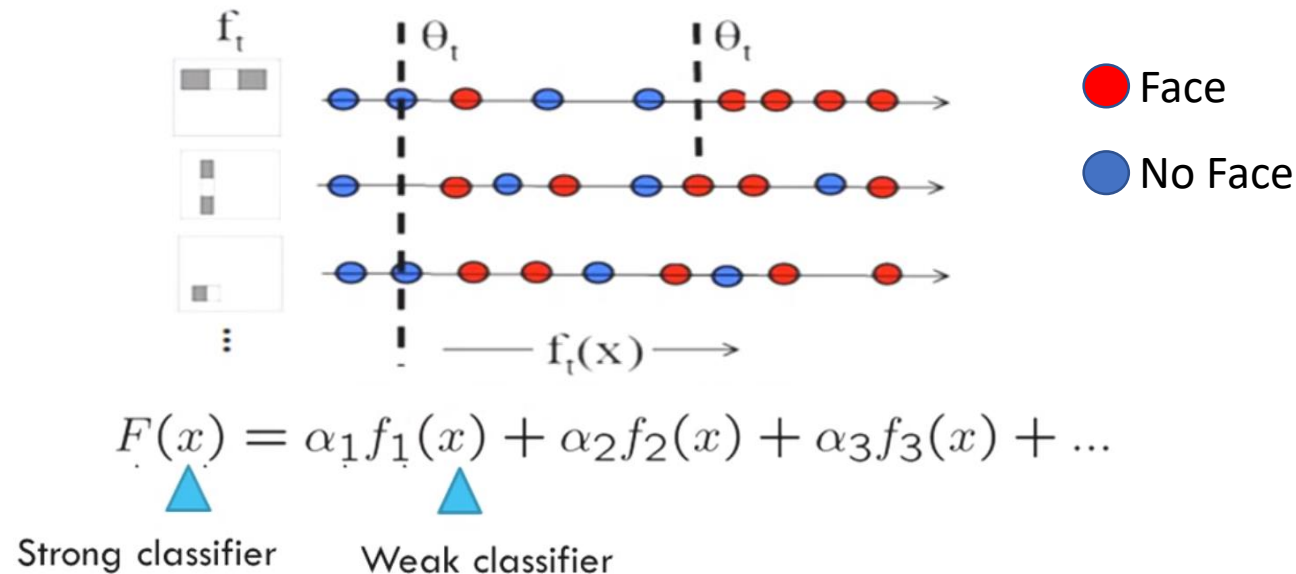
Integral Image

3	11	13	14
9	20	31	39
14	27	42	59
20	33	49	74

$$42 - 13 - 14 + 3 = 18$$

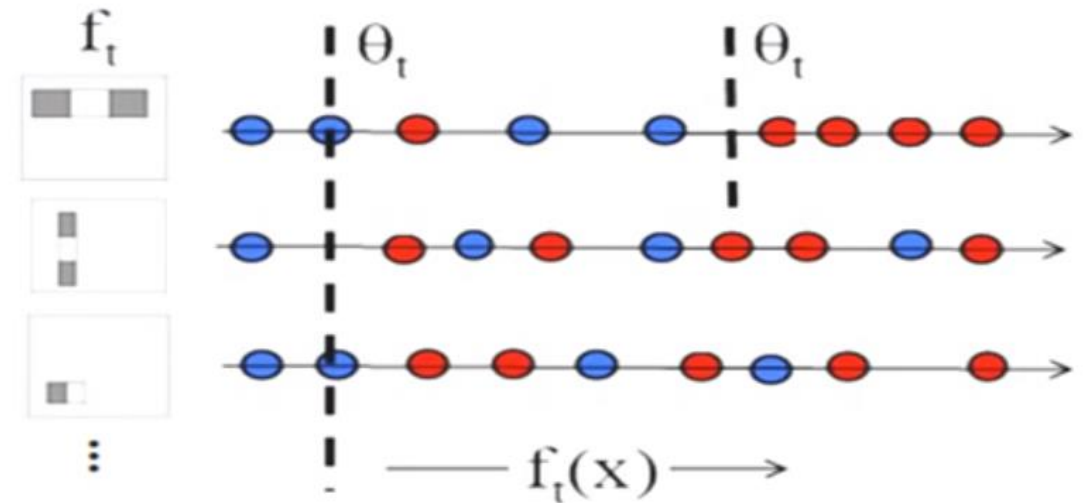
Viola Jones Face Detection - Adaboost

- There are so many relevant and irrelevant features
- More than 160000 feature for 24x24 window size.
- Adaboost classifier is used to reduce this number
- Find only the best features (6000 features) – 200 is enough for 95% accuracy



Viola Jones Face Detection - Adaboost

- Each classifier works on one feature
- Face or No Face
- Find threshold
- Increase the weight for misclassifications
- Recalculate it until error rate is low enough or
- A desired number of good features found.



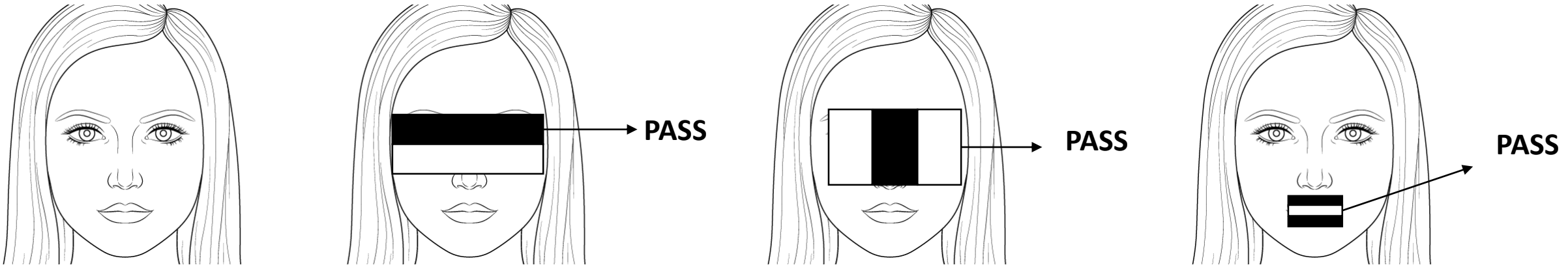
$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \dots$$

Strong classifier

Weak classifier

Viola Jones Face Detection - Cascading

- Try all different Haar features in all possible sizes and locations
- Cascade of Classifiers
- Each stage has different number of features
- If it passes one stage go to the next stage



Face Detection Demo

```
import numpy as np
import cv2

face_cascade = cv2.CascadeClassifier('C:\\OpenCV-3.3.1\\opencv\\build\\etc\\haarcascades\\haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('C:\\OpenCV-3.3.1\\opencv\\build\\etc\\haarcascades\\haarcascade_eye.xml')

img = cv2.imread('melih.jpg')

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)


















faces = face_cascade.detectMultiScale(gray, 1.3, 5)

for (x,y,w,h) in faces:
    cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]

    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color, (ex,ey), (ex+ew,ey+eh), (0,255,0),2)

cv2.imshow('img',img)

k = cv2.waitKey(0)
if k == 27:      # wait for ESC key to exit
    cv2.destroyAllWindows()
elif k == ord('s'): # wait for 's' key to save and exit
    cv2.imwrite('messigray.png',img)
    cv2.destroyAllWindows()
```

-  haarcascade_eye.xml
-  haarcascade_eye_tree_eyeglasses.xml
-  haarcascade_frontalcatface.xml
-  haarcascade_frontalcatface_extended.
-  haarcascade_frontalface_alt.xml
-  haarcascade_frontalface_alt_tree.xml
-  haarcascade_frontalface_alt2.xml
-  haarcascade_frontalface_default.xml
-  haarcascade_fullbody.xml
-  haarcascade_lefteye_2splits.xml
-  haarcascade_licence_plate_rus_16stagn...
-  haarcascade_lowerbody.xml
-  haarcascade_profileface.xml
-  haarcascade_righteye_2splits.xml
-  haarcascade_russian_plate_number.xml
-  haarcascade_smile.xml
-  haarcascade_upperbody.xml

Outline

- Introduction
- Motivation
- Steps to Create Panorama
- Results
- Questions

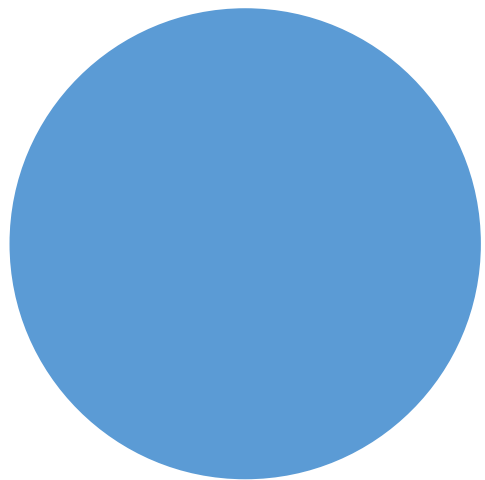


Image Panorama



Introduction

- Often one camera for object recognition is not enough
- Not all objects captured in an image
- Using two cameras to create a panorama and use that panoramic image for object detection

5 Steps to Make Panorama

**Step
1**

**Capture
Images**

**Step
2**

**Feature
Detection &
Matching**

**Step
3**

Align Images

**Step
4**

Blending

**Step
5**

Cropping

Capture Images

Step 1 – Capture Images



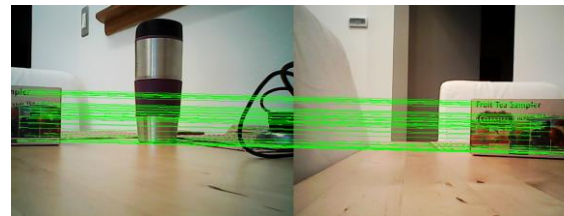
Create a mask size of 2 images



- Create a mask for the warped image

Step 2 - Feature Detection and Matching

Find Matches - SIFT

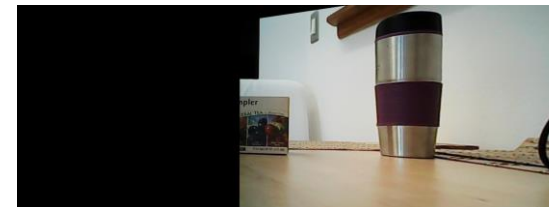
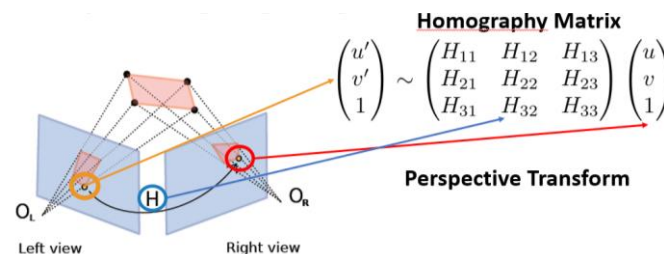


RANSAC



Calculate Homography and warp the image

Step 3 – Aligning Images



Step 4 Blending

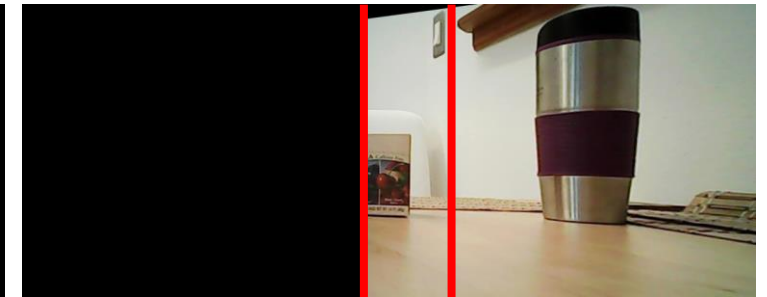


Overlap 20%

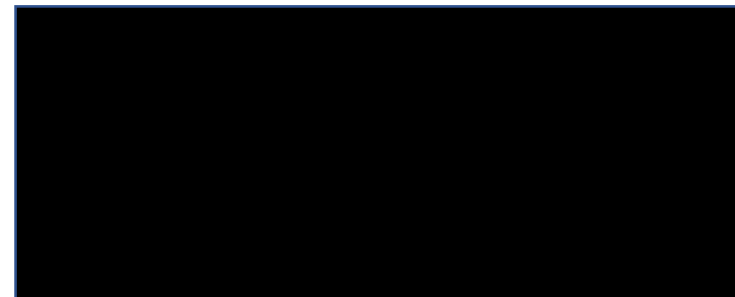
Overlap 20%

Preprocessing images

- Create a mask for input images

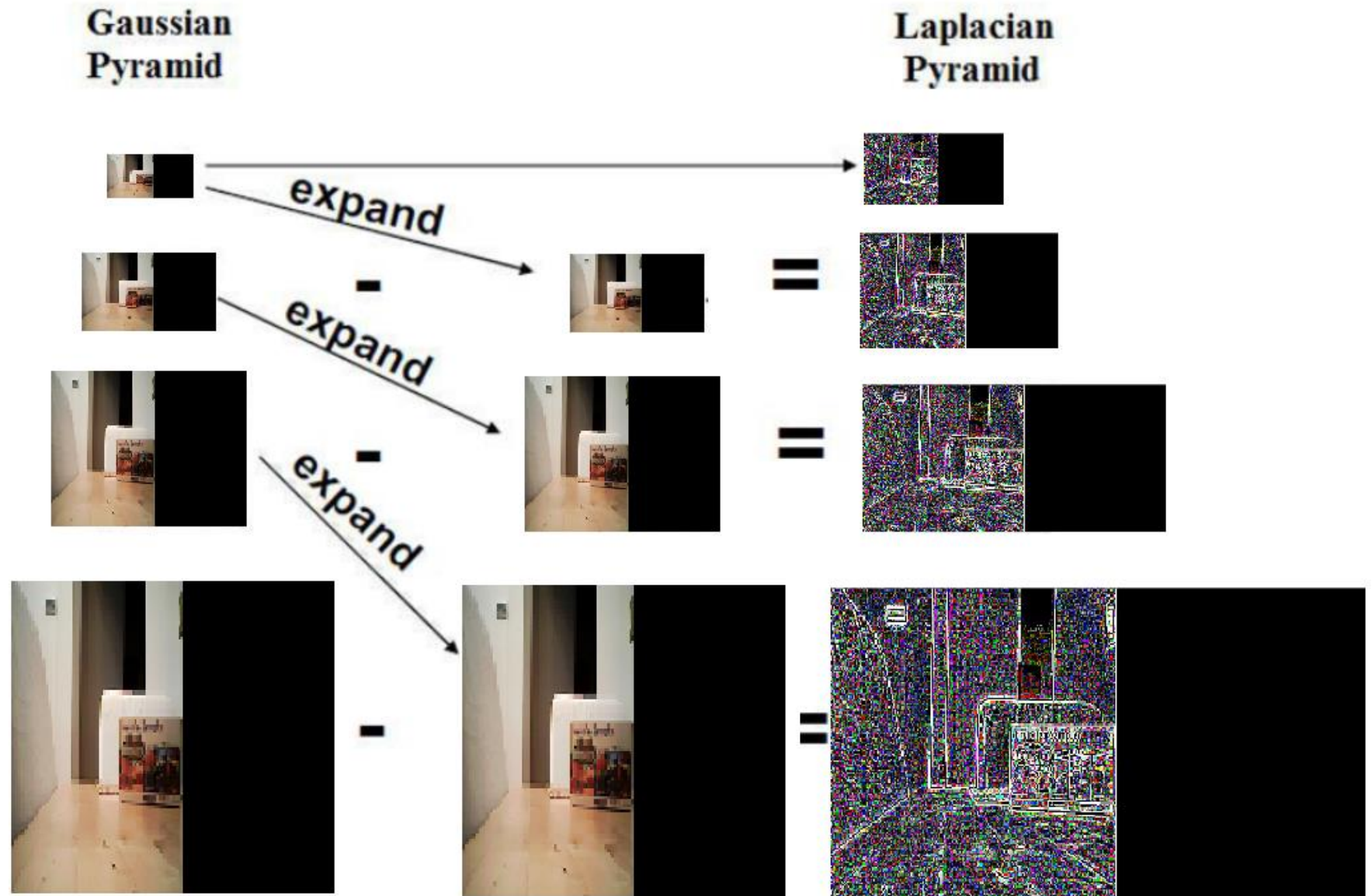


- Create a mask for the final image



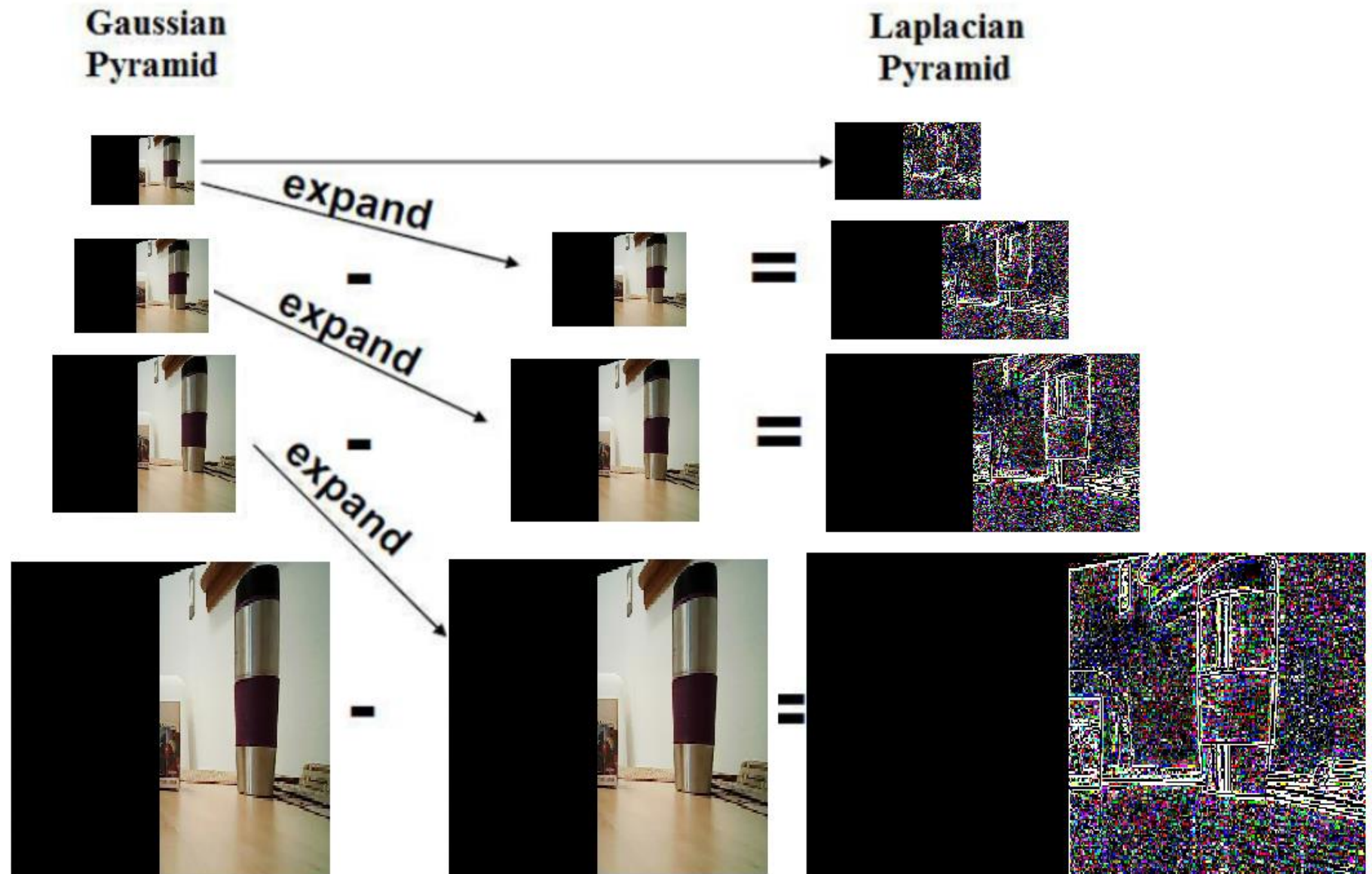
Step 4 Blending

- Build Laplacian pyramids LA

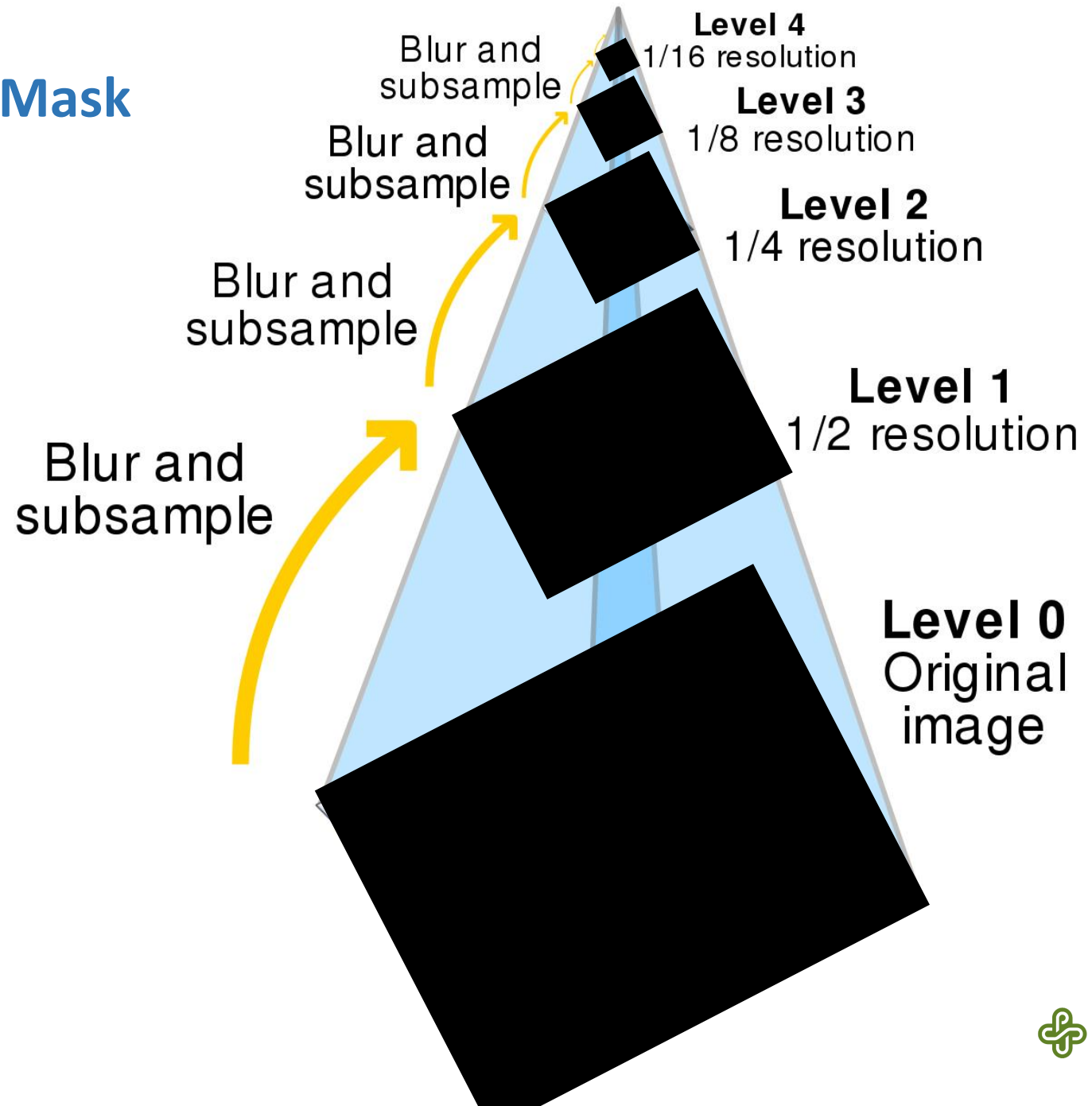


Step 4 Blending

- Build Laplacian pyramids LB



Gaussian Pyramid of the Mask



Blending: Form a combined pyramid from LA and LB

$$LS(i,j) = \text{GR}(I,j) * \text{LA}(I,j) + (1-\text{GR}(I,j)) * \text{LB}(I,j)$$

GR(I,j)

*

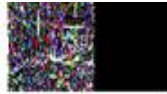
LA(I,j)

+

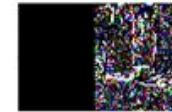
(1-GR(I,j))

*

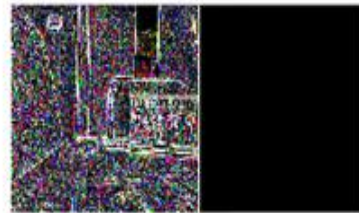
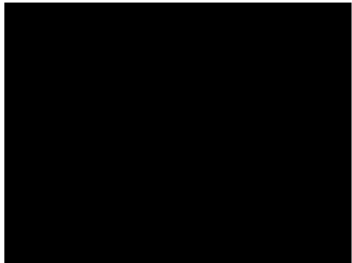
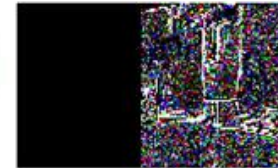
LB(I,j)



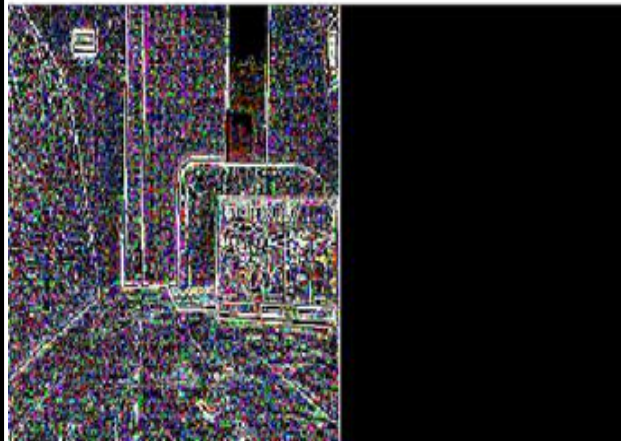
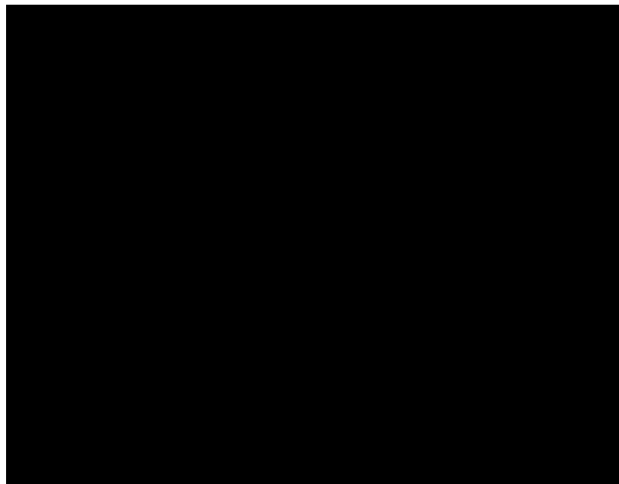
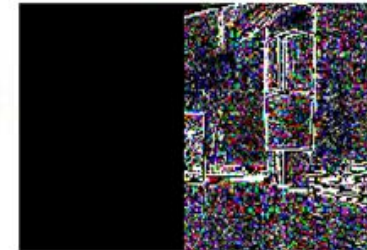
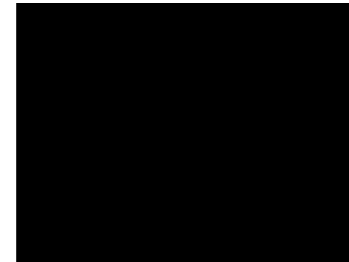
1-



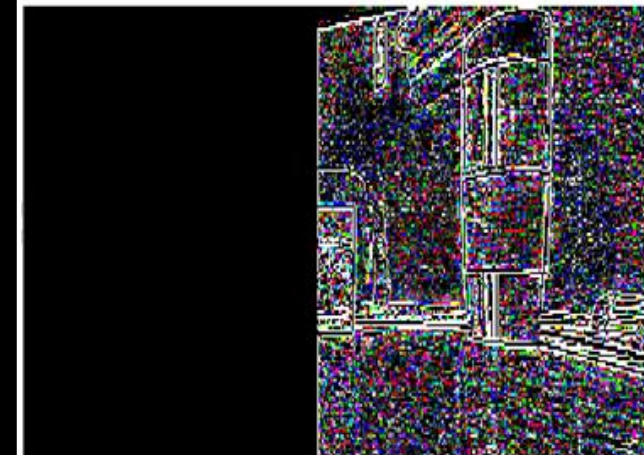
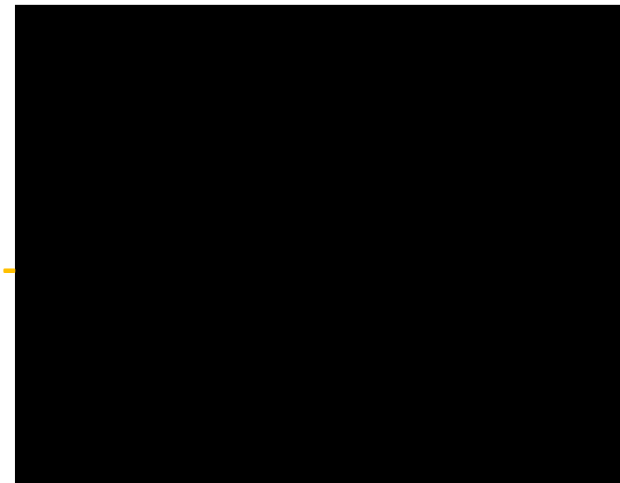
1-



1-

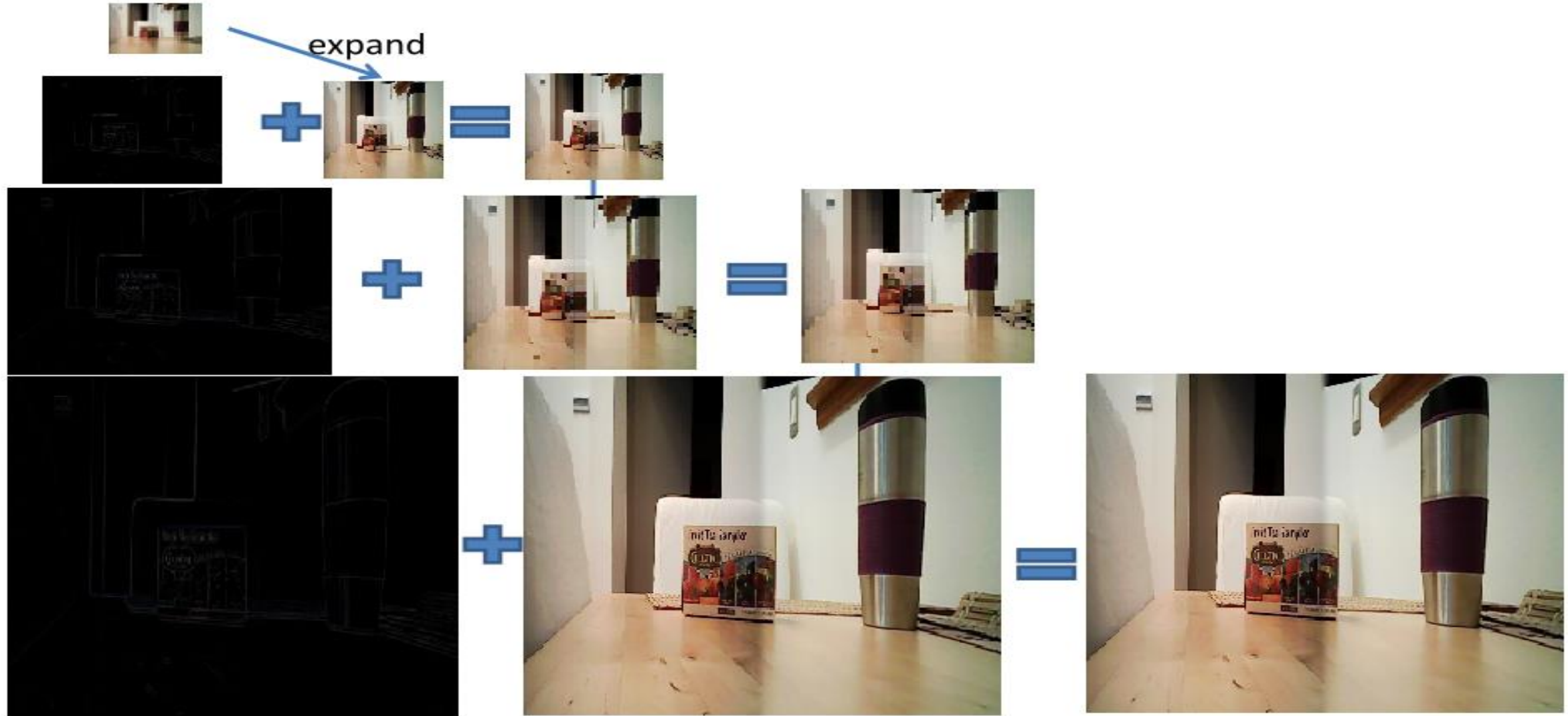


1-



Reconstruct LS pyramid to get the final blended image

LS(i,j)



Pyramid Blended Image

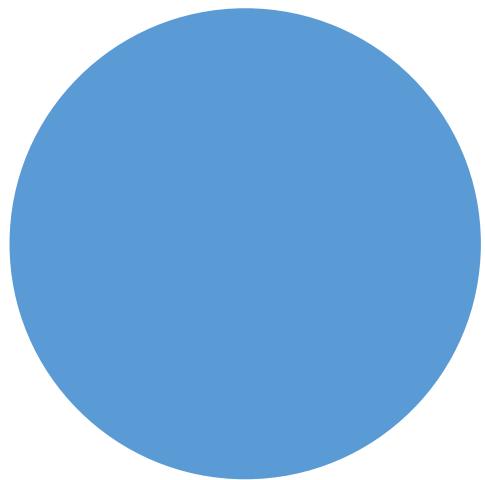


Step 5 – Cropping (Optional)



What if we don't use any blending





Results











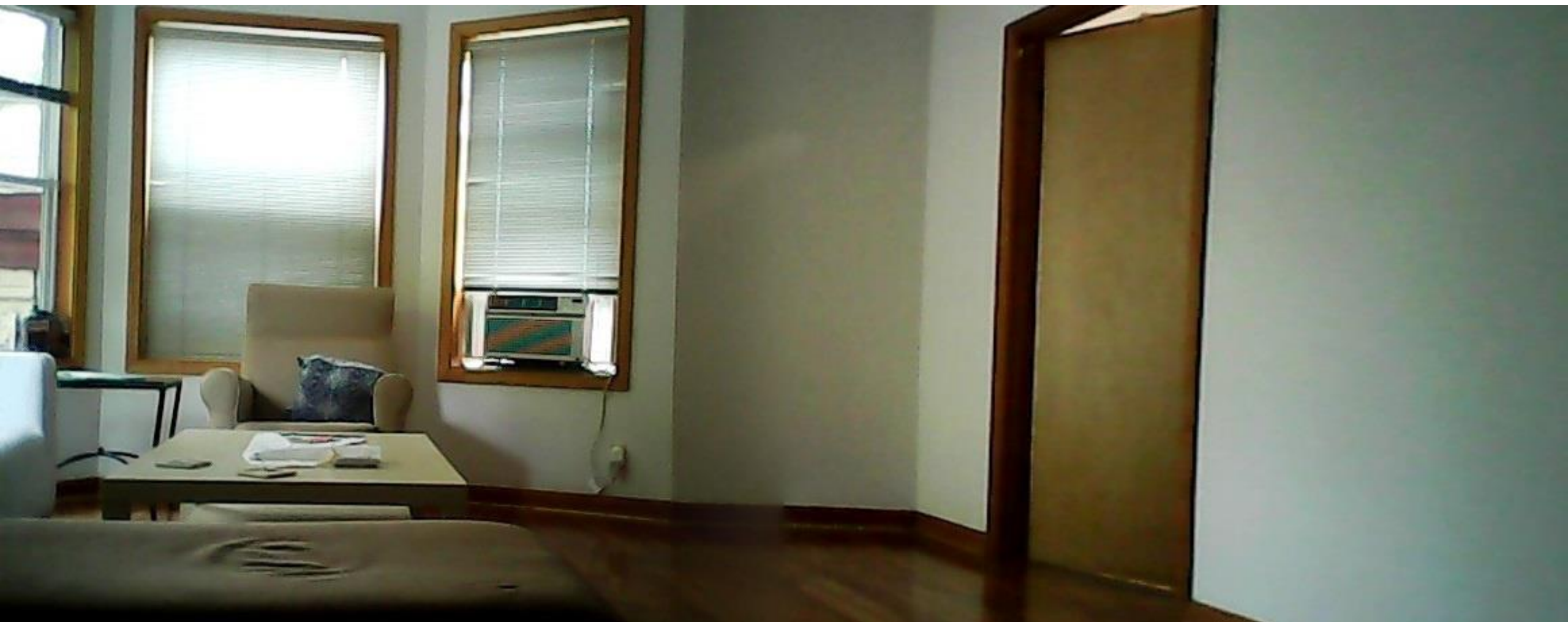












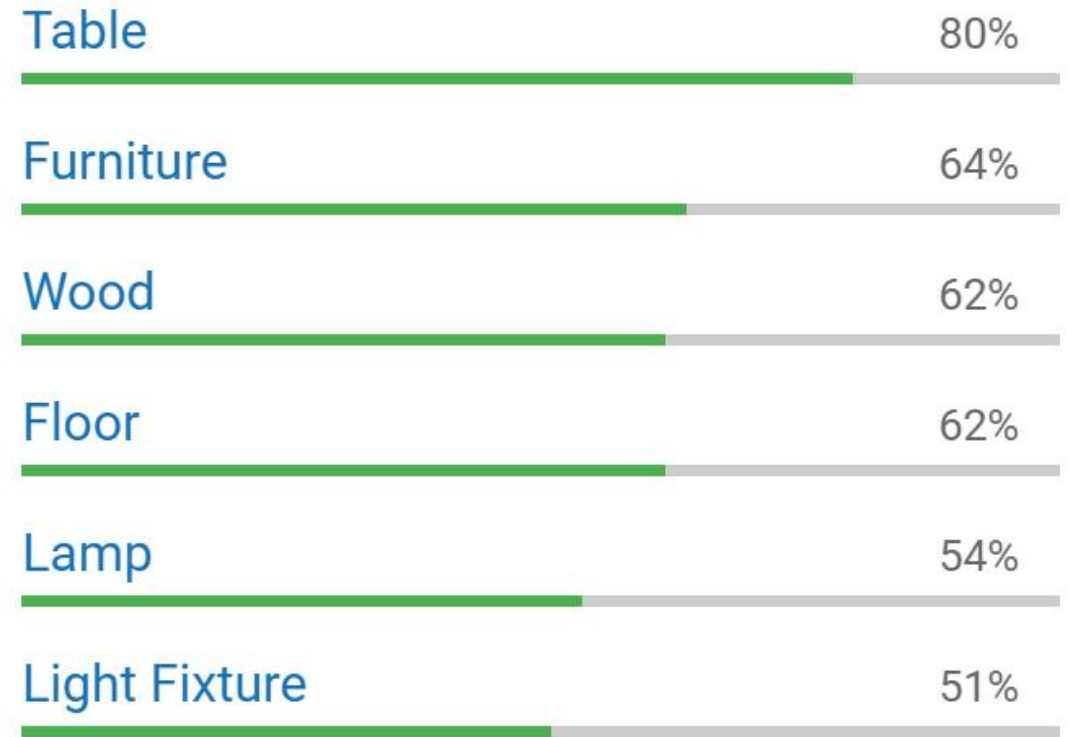
Object Detection



Google Cloud Vision API with Standard Image



r3.jpg

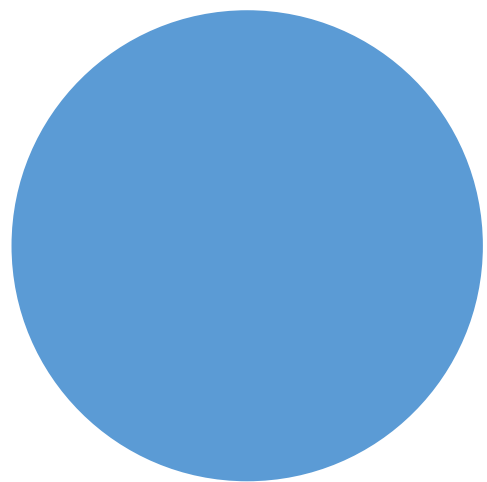


Google Cloud Vision API with Panoramic Image



result3.jpg

Table	83%
Cup	82%
Floor	81%
Coffee Cup	75%
Ceramic	75%
Flooring	71%
Tableware	68%
Furniture	65%
Glass	64%



Questions?

