2110433 - Computer Vision (2023/2)

Lab 4 - Color Processing

In this lab, we will learn to use color feature in different color spaces to extract useful information from images. This notebook includes both coding and written questions. Please hand in this notebook file with all outputs and your answer

Import OpenCV, Numpy and Matplotlib as always

In [87]:
```
import cv2
import math
import time
import threading
import numpy as np
import random as rng
from matplotlib import pyplot as plt
from ipywidgets import interact,interactive
import ipywidgets as widgets
from IPython.display import display, HTML,clear_output
import IPython.display
%matplotlib inline
```

Grayscale Image Thresholding

Thresholding is the simplest method of image segmentation. This non-linear operation converts a grayscale image into a binary image where the two colors (black/white) are assigned to pixels that are below or above the specified threshold.
Lena comes again! Can you adjust both sliders to segment lena's skin?

In [49]:
```
inputImage = cv2.imread("assets/lena_std.tif",cv2.IMREAD_GRAYSCALE)
def grayscaleThresholding(minValue,maxValue):
    thresholdImage = np.logical_and(inputImage > minValue, inputImage < maxV
    inputImageCopy = inputImage.copy()
    cv2.rectangle(inputImageCopy,(250,400),(340,500),255,3)
    cropRegion = inputImage[400:500,250:340]

    plt.figure(figsize=(10,10))
    plt.subplot(131)
    plt.title("Lena Image")
    plt.imshow(inputImageCopy, cmap='gray')

    plt.subplot(132)
    plt.title("Segmentation Mask")
    plt.imshow(thresholdImage, cmap='gray')

    plt.subplot(133)
    plt.title("Pixel Value Distribution")
    plt.hist(cropRegion,range=(0,255))
    plt.show()
interact(grayscaleThresholding, minValue=widgets.IntSlider(min=0,max=255,ste
```

```
interactive(children=(IntSlider(value=1, description='minValue', max=255), I
ntSlider(value=1, description='max…
```

Simple Image Segmentation using Color

As you can see from the above sample, only grayscale information is usually not enough to segment "things" from the images. In this section we will apply simple color segmentation on various colorspaces. The following block is code snippet which retrive image from your webcam and apply thresholding on BGR image using defined value.

In [50]:
```
bMin = 50; bMax = 170
gMin = 70; gMax = 180
rMin = 90; rMax = 220
cameraNo = 0
```

In [51]:
```
# You can press "Interupt Kernel Button to stop webcam"
inputStream = cv2.VideoCapture(cameraNo)
try:
    while True:
        _, videoFrameBGR = inputStream.read()
        if videoFrameBGR is not None:
            outputVideoFrameBGR = videoFrameBGR.copy()

            # Draw ROI
            cv2.rectangle(outputVideoFrameBGR,(100,100),(200,200),(0,255,0),
            # Cropped Region
            croppedRegion = videoFrameBGR[100:200,100:200,:]



            mask = cv2.inRange(videoFrameBGR,(bMin,gMin,rMin),(bMax,gMax,rMa
            mask = np.repeat(mask,3,axis=2)

            ## Draw Min/Max pixel value in BGR order on image
            cv2.putText(outputVideoFrameBGR,str(np.min(croppedRegion[:,:,0])
            cv2.putText(outputVideoFrameBGR,str(np.max(croppedRegion[:,:,0])

            outputVideoFrameBGR = np.hstack((outputVideoFrameBGR,mask))

            # Encode image as jpg numpy array
            _, buf = cv2.imencode(".jpg", outputVideoFrameBGR)
            # Draw result
            IPython.display.display(IPython.display.Image(data=buf))

            clear_output(wait=True)
        else:
            print("Cannot Open Webcam, hw problem?")
            break
```

```
except KeyboardInterrupt:
    print ("Stream stopped")
inputStream.release()
```

```
Stream stopped
```

Since the slider widget does not support for-loop webcam retrival method that we use, we may use build-in OpenCV GUI library to create a color range slider by using the following code. (The window name **"Color Segmentation"** will popup!)

In [76]:
```
def sliderCallback(x):
    pass
# Create a OpenCV Window
windowName = 'Color Segmentation'
cv2.namedWindow(windowName)
cv2.createTrackbar('bMin',windowName,0,255,sliderCallback)
cv2.createTrackbar('gMin',windowName,0,255,sliderCallback)
cv2.createTrackbar('rMin',windowName,0,255,sliderCallback)
cv2.createTrackbar('bMax',windowName,0,255,sliderCallback)
cv2.createTrackbar('gMax',windowName,0,255,sliderCallback)
cv2.createTrackbar('rMax',windowName,0,255,sliderCallback)

inputStream = cv2.VideoCapture(cameraNo)
try:
    while True:
        _, videoFrameBGR = inputStream.read()
        if videoFrameBGR is not None:


            bMin = cv2.getTrackbarPos('bMin',windowName)
            gMin = cv2.getTrackbarPos('gMin',windowName)
            rMin = cv2.getTrackbarPos('rMin',windowName)

            bMax = cv2.getTrackbarPos('bMax',windowName)
            gMax = cv2.getTrackbarPos('gMax',windowName)
            rMax = cv2.getTrackbarPos('rMax',windowName)

            mask = cv2.inRange(videoFrameBGR,(bMin,gMin,rMin),(bMax,gMax,rMa
            mask = np.repeat(mask,3,axis=2)
            outputVideoFrameBGR = videoFrameBGR.copy()
            outputVideoFrameBGR = np.hstack((outputVideoFrameBGR,mask))

            cv2.imshow(windowName,outputVideoFrameBGR)
            if cv2.waitKey(1) == ord('q'):
                cv2.destroyAllWindows()
                break
        else:
            print("Cannot Open Webcam, hw problem?")
            break
except KeyboardInterrupt:
    print ("Stream stopped")
inputStream.release()
cv2.destroyAllWindows()
```

```
Stream stopped
```

OpenCV supports many well-known colorspaces. You can apply the colorspace transformation by using cv2.cvtColor and see the list of suppoted transformation flags here. Try tp apply color segmention on any object in other colorspace **(NOT BGR!!)** by fill the following block.

In [114…

```
### FILL HERE ###

# >>>>>>>>>>>>>>>>>>>>    YCrCb    <<<<<<<<<<<<<<<<<<<<

def sliderCallback(x):
    pass
# Create a OpenCV Window
windowName = 'Color Segmentation'
cv2.namedWindow(windowName)
cv2.createTrackbar('YMin',windowName,0,255,sliderCallback)
cv2.createTrackbar('CbMin',windowName,0,255,sliderCallback)
cv2.createTrackbar('CrMin',windowName,0,255,sliderCallback)
cv2.createTrackbar('YMax',windowName,0,255,sliderCallback)
cv2.createTrackbar('CbMax',windowName,0,255,sliderCallback)
cv2.createTrackbar('CrMax',windowName,0,255,sliderCallback)

inputStream = cv2.VideoCapture(cameraNo)
try:
    while True:
        _, videoFrameBGR = inputStream.read()
        if videoFrameBGR is not None:
            videoFrameYCbCr = cv2.cvtColor(videoFrameBGR,cv2.COLOR_BGR2YCrCb

            YMin = cv2.getTrackbarPos('YMin',windowName)
            CbMin = cv2.getTrackbarPos('CbMin',windowName)
            CrMin = cv2.getTrackbarPos('CrMin',windowName)

            YMax = cv2.getTrackbarPos('YMax',windowName)
            CbMax = cv2.getTrackbarPos('CbMax',windowName)
            CrMax = cv2.getTrackbarPos('CrMax',windowName)

            # mask = cv2.inRange(videoFrameBGR,(bMin,gMin,rMin),(bMax,gMax,r
            mask = cv2.inRange(videoFrameYCbCr,(YMin, CbMin, CrMin), (YMax,
            mask = np.repeat(mask,3,axis=2)
            videoFrameBGR = cv2.cvtColor(videoFrameYCbCr,cv2.COLOR_YCrCb2BGR
            outputVideoFrameBGR = videoFrameBGR.copy()
            outputVideoFrameBGR = np.hstack((outputVideoFrameBGR,mask))

            cv2.imshow(windowName,outputVideoFrameBGR)
            if cv2.waitKey(1) == ord('q'):
                cv2.destroyAllWindows()
                break
        else:
            print("Cannot Open Webcam, hw problem?")
            break
except KeyboardInterrupt:
    print ("Stream stopped")
inputStream.release()
cv2.destroyAllWindows()
```

```
Stream stopped
```

In [77]:
```
# >>>>>>>>>>>>>>>>>>>>    HSV    <<<<<<<<<<<<<<<<<<<<

def sliderCallback(x):
    pass
# Create a OpenCV Window
windowName = 'Color Segmentation'
cv2.namedWindow(windowName)
cv2.createTrackbar('hMin',windowName,0,255,sliderCallback)
cv2.createTrackbar('sMin',windowName,0,255,sliderCallback)
cv2.createTrackbar('vMin',windowName,0,255,sliderCallback)
cv2.createTrackbar('hMax',windowName,0,255,sliderCallback)
cv2.createTrackbar('sMax',windowName,0,255,sliderCallback)
cv2.createTrackbar('vMax',windowName,0,255,sliderCallback)

inputStream = cv2.VideoCapture(cameraNo)
try:
    while True:
        _, videoFrameBGR = inputStream.read()
        if videoFrameBGR is not None:
            videoFrameHSV = cv2.cvtColor(videoFrameBGR,cv2.COLOR_BGR2HSV)

            hMin = cv2.getTrackbarPos('hMin',windowName)
            sMin = cv2.getTrackbarPos('sMin',windowName)
            vMin = cv2.getTrackbarPos('vMin',windowName)

            hMax = cv2.getTrackbarPos('hMax',windowName)
            sMax = cv2.getTrackbarPos('sMax',windowName)
            vMax = cv2.getTrackbarPos('vMax',windowName)

            mask = cv2.inRange(videoFrameHSV,(hMin,sMin,vMin),(hMax,sMax,vMa
            mask = np.repeat(mask,3,axis=2)
            videoFrameHSV = cv2.cvtColor(videoFrameHSV,cv2.COLOR_HSV2BGR)
            outputVideoFrameHSV = videoFrameHSV.copy()
            outputVideoFrameHSV = np.hstack((outputVideoFrameHSV,mask))

            cv2.imshow(windowName,outputVideoFrameHSV)
            if cv2.waitKey(1) == ord('q'):
                cv2.destroyAllWindows()
                break
        else:
            print("Cannot Open Webcam, hw problem?")
            break
except KeyboardInterrupt:
    print ("Stream stopped")
inputStream.release()
cv2.destroyAllWindows()
```

```
Stream stopped
```

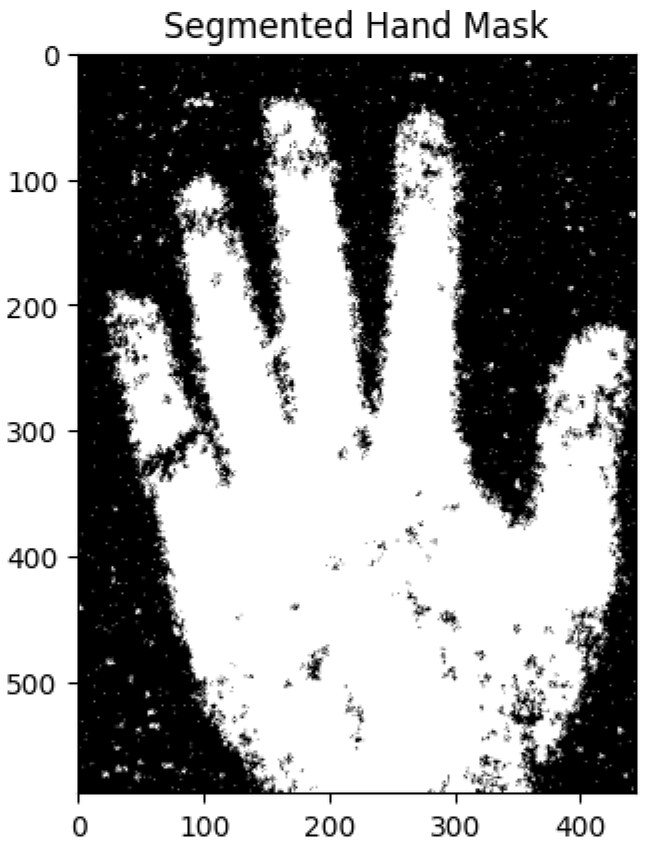
Morphological Transformations

The field of mathematical morphology contributes a wide range of operators to image processing, all based around a simple mathematical concepts from set theory.

Morphological transformations are the operations based on the image shape employed on binay images. This operation needs needs two inputs, one is binary image, second one is called **structuring element or kernel** which decides the operation output. You can design the kernel to suit your application needs. Two basic morphological operators are Erosion and Dilation

The following mask image is segmented by using color information. You can see that there are some hand's pixels which are not connect into a perfect hand shape. We can correct these by using the basic morphological operaters.

In [44]:
```python
handMask = cv2.imread('assets/Lab4-SegmentedHand.png',cv2.IMREAD_GRAYSCALE)
plt.title('Segmented Hand Mask')
plt.imshow(handMask,cmap='gray')
plt.show()
```



In [45]:
```python
def openAndCloseMorph(kernelSize,kernelShape, morphType):
    kernel = cv2.getStructuringElement(kernelShape,(kernelSize,kernelSize))

    outputImage = handMask.copy()

    if morphType == 'erode':
        outputImage = cv2.erode(outputImage,kernel,iterations = 1)
    else:
        outputImage = cv2.dilate(outputImage,kernel,iterations = 1)

    plt.figure(figsize=(5,5))
    plt.imshow(outputImage, cmap='gray')
    plt.show()

    print('Morphology Kernel Shape:')
    display(kernel)

interact(openAndCloseMorph, kernelSize=widgets.IntSlider(min=1,max=11,step=1
         kernelShape=widgets.Dropdown(
                        options=[cv2.MORPH_RECT,cv2.MORPH_ELLIPSE, cv2.MORPH
                        value=cv2.MORPH_RECT,
                        description='kernelShape:',
                        disabled=False),
        morphType=widgets.Dropdown(
                        options=['erode','dilate'],
                        value='erode',
                        description='Morph Type:',
                        disabled=False)
);
```

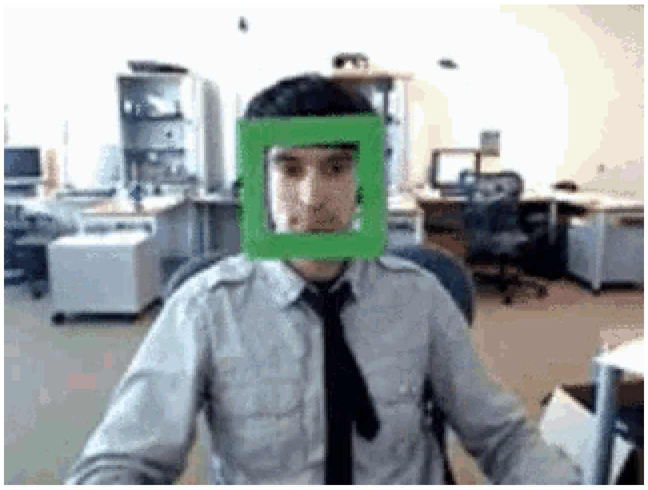
```
interactive(children=(IntSlider(value=1, description='kernelSize', max=11, m
in=1), Dropdown(description='kerne…
```

This page shows a good morphological operation exmple, try to write an interactive visualization like the above sample on **Opening and Closing** operations. See the output results by yourself.

In [46]:
```
### FILL HERE ###
def openAndCloseMorph(kernelSize,kernelShape, morphType):
    kernel = cv2.getStructuringElement(kernelShape,(kernelSize,kernelSize))

    outputImage = handMask.copy()

    if morphType == 'erode, dilate':
        outputImage1 = cv2.erode(outputImage,kernel,iterations = 1)
        outputImage2 = cv2.dilate(outputImage,kernel,iterations = 1)
    elif morphType == 'opening, closing':
        outputImage1 = cv2.morphologyEx(outputImage, cv2.MORPH_OPEN, kernel)
        outputImage2 = cv2.morphologyEx(outputImage, cv2.MORPH_CLOSE, kernel
    elif morphType == 'opening -> closing':
        outputImage1 = cv2.morphologyEx(outputImage, cv2.MORPH_OPEN, kernel)
        outputImage2 = cv2.morphologyEx(outputImage1, cv2.MORPH_CLOSE, kerne
    else:
        outputImage1 = cv2.morphologyEx(outputImage, cv2.MORPH_CLOSE, kernel
        outputImage2 = cv2.morphologyEx(outputImage1, cv2.MORPH_OPEN, kernel
    plt.figure(figsize=(10, 5))
    plt.subplot(121)
    plt.imshow(outputImage1, cmap='gray')
    plt.subplot(122)
    plt.imshow(outputImage2, cmap='gray')
    plt.show()

    print('Morphology Kernel Shape:')
    display(kernel)

interact(openAndCloseMorph, kernelSize=widgets.IntSlider(min=1,max=11,step=1
         kernelShape=widgets.Dropdown(
                        options=[cv2.MORPH_RECT,cv2.MORPH_ELLIPSE, cv2.MORPH
                        value=cv2.MORPH_RECT,
                        description='kernelShape:',
                        disabled=False),
        morphType=widgets.Dropdown(
                        options=['opening, closing', 'erode, dilate', 'openi
                        value='opening, closing',
                        description='Morph Type:',
                        disabled=False)
);
#################
```

```
interactive(children=(IntSlider(value=1, description='kernelSize', max=11, m
in=1), Dropdown(description='kerne…
```

Assignment 1 - Color Based Face Detector


By using the knowledge from lecture 1-4, you should be able to write your own simple color based face detector. Use the above code snippets to help you write it. The output should be a code which retrive video feed from **your webcam** and draw bounding boxes around detected faces. Write the detection results into video file and hand in with this notebook. There should be **two video sequences**, in good lighting and other lighting condition. The output video should show robustness of your designed alogorithm. (Optional) You will get extra points if you can use **same parameters** for both sequences.

Basic Guidance:

1. Create a "face color segmentation mask" using your choice colorspace.
2. Filter out the outlier pixel!
3. Categorize each connected component into group by using cv2.findContours (from Lab 3)
4. Find the bounding box which can enclose those connect components by cv2.boundingRect

Hints:

- From today lecture, how do to discard noise/fill small hole from color segmentation mask output?
- Since this is a color-based problem, you can use old knowledge from lecture 1-3 to improve segmentation result by apply **?** on input image
- You can use some specific threshold based on shape properties or simple morphological operations to keep only potential contours
- To achieve a better result for both lighting conditions, you may need to apply some data analysis on the **region of interest** by plotting each channel value and see their data distributions.
- Internet is your friend. You can search for relavent research papers and use their algorithms/implementations, but you must **give proper credits** by citing them in this notebook.

In [ ]:
```
### Describe how your algorithm work here (Thai or English). You can provide
'''
1. Convert each frame BRG color space to YCrCb color space because YCrCb is
2. Tune the YCrCb to detect the skin and work on both low and high light con
3. Use Closing Morphology to fill the hole inside then Opening Morphology to
4. Find Contours and choose the biggest to be face area because another smal
5. Use vertical Opening Morphology with relative size to face to remove ears
6. (try to remove neck by the output of 5. make neck width equally in that p
7. Find Contours again and choose the biggest then draw the rectangle boundi
'''
```

In [118…
```
idx_video = 4
video_buffer_path = 'Output/Lab4/A1/buffer/out'+str(idx_video)+'.mp4'
video_output_path = 'Output/Lab4/A1/BW/out'+str(idx_video)+'.mp4'
video_contour_path = 'Output/Lab4/A1/contour/out'+str(idx_video)+'.mp4'
video_sobel_path = 'Output/Lab4/A1/sobel/out'+str(idx_video)+'.mp4'
```

In [123…
```
### FILL HERE ###
inputStream = cv2.VideoCapture(0)
inputFPS = int(30)
inputWidth = int(inputStream.get(cv2.CAP_PROP_FRAME_WIDTH))
inputHeight = int(inputStream.get(cv2.CAP_PROP_FRAME_HEIGHT))
outputStream = cv2.VideoWriter(video_buffer_path,
                               cv2.VideoWriter_fourcc('x', '2', '6', '4'),
                               inputFPS, (inputWidth, inputHeight))
try:
    while True:
        _, videoFrame = inputStream.read()
        if videoFrame is not None:

            ### Do something with videoFrame Here ###
            ### FILL HERE ###
            outputStream.write(videoFrame)
            #################

            # Encode image as jpg numpy array
            _, buf = cv2.imencode(".jpg", videoFrame)
            # Draw result
            IPython.display.display(IPython.display.Image(data=buf))
            # Discard old output
            clear_output(wait=True)
        else:
            print("Cannot Open Webcam, hw problem?")
            break

except KeyboardInterrupt:
    print ("Stream stopped")
inputStream.release()
outputStream.release()
```

```
Stream stopped
```

In [124…

```
inputStream = cv2.VideoCapture(video_buffer_path)
output = []
output_contour = []
output_sobel = []
def findContour(pic):
    contour, _ = cv2.findContours(pic, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_S
    idx, ma = 0, 0
    for i in range(len(contour)):
        area = cv2.contourArea(contour[i])
        if area > ma:
            ma = area
            idx = i
    x, y, w, h = cv2.boundingRect(contour[idx])
    return x, y, w, h

def findHorizontalLine(arr):
    a1 = np.array(arr) > 1
    d = np.diff(np.concatenate([np.repeat(np.array([False])[np.newaxis, :],

    a = np.full((a1.shape[0], 2), np.nan)
    for i in range(a1.shape[0]):
        aa = np.where(d[i])[0]
        aa = aa[~(aa > a1.shape[1])].reshape(-1, 2)
        if aa.size == 0:
            continue
        aa2 = np.diff(aa, axis=1).reshape(-1).argmax()
        a[i] = aa[aa2]
    uni = np.unique(a[~np.isnan(a).any(axis=1)], axis=0, return_counts=True)
    ans = uni[0][uni[1] > 20]
    # print(uni)
    ch = 0
    key = 0
    for i in range(a.shape[0]-1,-1,-1):
        if a[i] in ans:
            if ch == 0:
                key = i
                ch = 1
            elif a[i][0] != a[key][0] or a[i][1] != a[key][1]:
                continue
            arr[i] = np.full(arr.shape[1], 255)
    return arr.copy()
```

```
while True:
    _, frame = inputStream.read()
    if frame is not None:
        videoFrameYCbCr = cv2.cvtColor(frame,cv2.COLOR_BGR2YCrCb)
        mask = cv2.inRange(videoFrameYCbCr,(0, 137, 92), (236, 165, 141))[:,
        s = 15
        kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(s,s))
        mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
        mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
        x, y, w, h = findContour(mask)
        s = int(h//5)
        kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(1,s))
        mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
        # mask = findHorizontalLine(mask)

        frameGray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        sobelX = cv2.Sobel(frameGray, cv2.CV_64F, 1, 0, ksize=3)
        sobelX = np.uint8(np.absolute(sobelX))
        sobelY = cv2.Sobel(frameGray, cv2.CV_64F, 0, 1, ksize=3)
        sobelY = np.uint8(np.absolute(sobelY))
        # sobel = cv2.bitwise_or(sobelX, sobelY)
        sobel = sobelY
        sobel = np.reshape(sobel,(sobel.shape[0],sobel.shape[1],1))
        sobel = np.repeat(sobel,3,axis=2)
        output_sobel.append(sobel)


        mask = np.reshape(mask,(mask.shape[0],mask.shape[1],1))
        mask = np.repeat(mask,3,axis=2)
        output.append(mask)

        framee = frame.copy()
        x, y, w, h = findContour(mask[:,:,0])
        h -= h//5
        cv2.rectangle(framee, (x, y), (x+w, y+h), (0, 255, 0), 2)
        output_contour.append(framee)
    else:
        break
inputStream.release()
outputStream = cv2.VideoWriter(video_output_path,
                               cv2.VideoWriter_fourcc('x', '2', '6', '4'),
                               30, (inputWidth, inputHeight))
outputStream2 = cv2.VideoWriter(video_contour_path,
                                cv2.VideoWriter_fourcc('x', '2', '6', '4'),
                                30, (inputWidth, inputHeight))
outputStream3 = cv2.VideoWriter(video_sobel_path,
                                   cv2.VideoWriter_fourcc('x', '2', '6', '4
                                   30, (inputWidth, inputHeight))
for i in range(len(output)):
    outputStream.write(output[i])
    outputStream2.write(output_contour[i])
    outputStream3.write(output_sobel[i])
outputStream.release()
outputStream2.release()
outputStream3.release()
```

Assignment 2 - Invisibility Cloak

By using the knowledge from lecture 1-4, you should be able to mimic a invisibility cloak from famous Harry Potter franchise by using color segmentation

Basic Guidance:

1. Create a "invisibility cloak color segmentation mask" using your choice colorspace.
2. Filter out the outlier pixel using some specify (you can think by your own!) criteria.
3. Replace each invisible cloak area with store static background image.
4. Make a short video clip to demonstrate/show your algorithm

Hints:

- From today lecture, how do to discard noise/fill small hole from color segmentation mask output?
- Since this is a color-based problem, you can use old knowledge from lecture 1-3 to improve segmentation result by apply **?** on input image
- Internet is your friend. You can search for relavent research papers and use their algorithm, but you must **give proper credits** by citing them in this notebook.

In [79]:
```
idx_video = 1
video_buffer_path = 'Output/Lab4/A2/buffer/out'+str(idx_video)+'.mp4'
video_mask_path = 'Output/Lab4/A2/mask/out'+str(idx_video)+'.mp4'
video_output_path = 'Output/Lab4/A2/output/out'+str(idx_video)+'.mp4'
```

In [99]:
```
inputStream = cv2.VideoCapture(0)
inputFPS = int(30)
inputWidth = int(inputStream.get(cv2.CAP_PROP_FRAME_WIDTH))
inputHeight = int(inputStream.get(cv2.CAP_PROP_FRAME_HEIGHT))
outputStream = cv2.VideoWriter(video_buffer_path,
                               cv2.VideoWriter_fourcc('x', '2', '6', '4'),
                               inputFPS, (inputWidth, inputHeight))
time.sleep(2)
try:
    while True:
        _, videoFrame = inputStream.read()
        if videoFrame is not None:

            ### Do something with videoFrame Here ###
            ### FILL HERE ###
            outputStream.write(videoFrame)
            #################

            # Encode image as jpg numpy array
            _, buf = cv2.imencode(".jpg", videoFrame)
            # Draw result
            IPython.display.display(IPython.display.Image(data=buf))
            # Discard old output
            clear_output(wait=True)
        else:
            print("Cannot Open Webcam, hw problem?")
            break

except KeyboardInterrupt:
    print ("Stream stopped")
inputStream.release()
outputStream.release()
```

```
Stream stopped
```

In [101…

```
inputStream = cv2.VideoCapture(video_buffer_path)
output_mask = []
output = []
ch = 0
background = None

while True:
    _, frame = inputStream.read()
    if frame is not None:
        if ch == 0:
            background = frame
            ch = 1
        videoFrameYCbCr = cv2.cvtColor(frame,cv2.COLOR_BGR2YCrCb)
        mask = cv2.inRange(videoFrameYCbCr,(0, 106, 136), (255, 133, 175))[:
        s = 15
        kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(s,s))
        mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
        mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
        mask = cv2.morphologyEx(mask, cv2.MORPH_DILATE, kernel)

        mask = np.reshape(mask,(mask.shape[0],mask.shape[1],1))
        mask = np.repeat(mask,3,axis=2)
        output_mask.append(mask)

        outputFrame = np.where(mask == 0, frame, background)
        output.append(outputFrame)

    else:
        break
inputStream.release()
outputStream = cv2.VideoWriter(video_mask_path,
                               cv2.VideoWriter_fourcc('x', '2', '6', '4'),
                               30, (inputWidth, inputHeight))
outputStream2 = cv2.VideoWriter(video_output_path,
                                 cv2.VideoWriter_fourcc('x', '2', '6', '4'),
                                 30, (inputWidth, inputHeight))

for i in range(len(output_mask)):
    outputStream.write(output_mask[i])
    outputStream2.write(output[i])
outputStream.release()
outputStream2.release()
```