# Assignment 2: OpenCV

Ana Maria Martinez Sidera
Department of Computer Science
Illinois Institute of Technology

October 9, 2017

## 1. Problem statement

The program should load an image by either reading it from a file or capturing it directly from a camera. When the user presses a key perform the operation corresponding to the key on the original image. The program should satisfy the following specifications:

1. The image to be processed by the program should be either read from a file or captured directly from a camera. If a file name is specified in the command line, the image should be read from it. Otherwise the program should attempt to capture an image from a camera. When capturing an image from the camera, continue to capture and process images continuously.

2. The read image should be read as a 3 channel color image.

3. The program should work for any size image. Make sure to test it on different size images.

4. Special keys on the keyboard should be used to modify the displayed image as follows:

   (a) 'i' - reload the original image

   (b) 'w' - save the current image into the file 'out.jpg'

   (c) 'g' - convert the image to grayscale using the openCV conversion function.

   (d) 'G' - convert the image to grayscale using your implementation of conversion function.

   (e) 'c' - cycle through the color channels of the image showing a different channel every time the key is pressed

   (f) 's' - convert the image to grayscale and smooth it using the openCV function. Use a track bar to control the amount of smoothing.

   (g) 'S' - convert the image to grayscale and smooth it using your function which should perform convolution with a suitable filter. Use a track bar to control the amount of smoothing.

   (h) 'd' - downsample the image by a factor of 2 whitout smoothing

   (i) 'D' - downsample the image by a factor of 2 with smoothing

   (j) 'x' - convert the image to grayscale and perform convolution with a x derivative filter. Normalize the obtained values to the range [0,255]

   (k) 'y' - convert the image to grayscale and perform convolution with a y derivative filter. Normalize the obtained values to the range [0,255]

   (l) 'm' - show the magnitude of the gradient normalized to the range [0,255]. The gradient is computed based on the x and y derivatives of the image.

(m) 'p' - convert the image to grayscale and plot the gradient vectors of the image every N pixels and let the plotted gradient vectors have a length of K. Use a track bar to control N. Plot the vectors as short line segments of length K.

(n) 'r' - convert the image to grayscale and rotate it using an angle of $\theta$ degrees. Use a track bar to control the rotation angle. The rotation of the image should be performed using a inverse map so there are no holes in it.

(o) 'h' - Display a short description of the program, its command line arguments, and the keys it supports.

## 2. Proposed solution

Provide a description of the algorithm you are implementing. Use references as needed

1. The first thing we have to do in this problem is to know if the user wants an image from a file or to take a picture with a camera. For this problem, the program print the following text on the screen: *"Enter a file name or press enter if you want a camera image:"*. The program will safe the input text in the variable *nb* the name of the file that the user wants.

```
def main():
    global nb
    nb = input('Enter a file name or press enter if you want a camera image: ')
```

2. The program pass through the function *reload()* the input above, that will check the value of the variable *nb*. The function can have three different results:

   (a) The user has a wrong file name and there isn't that image in the same folder. The program will give another opportunity to the user to enter a right file name or press Enter.

   (b) The user has correctly entered the file name and the program will uploaded the image and work with it.

   (c) The user has pressed *Enter* without any value and the variable *nb* is null, so the program will take the image of a camera. In this case, the camera will be the computer camera.

```
def readimage(name):
    if len(sys.argv)<2:
        img = cv2.imread(name)
    return img

def capturing():
    cap = cv2.VideoCapture(0)
    retval,image=cap.read()
    return image

def reload(nb):
    if nb!=0 and os.path.isfile(nb):
        imagen = readimage(nb)
```

```
    elif len(nb) > 1 and os.path.isfile(nb)==False:
        print ('There is no image with that name. Try again.')
        imagen=0
        main()
    elif len(nb) < 1:
        imagen=capturing()
    cv2.imshow('CS_512',imagen)
    return imagen
```

3. The program have to read the letters of the keyboard that the user is pressing. For each option, the program will call the function for each choice and perform the transformation required to the original image.

```
while True:
    key=cv2.waitKey()
    if key == ord('i'):
        imagen_new = reload(nb)
    elif key == ord('w'):
        writing(imagen_new)
    elif key == ord('g'):
        imagen_new = converting(reload(nb))
    elif key == ord('G'):
        imagen_new = convertinggray(reload(nb))
    elif key == ord('c'):
        imagen_new, counter = cycle(nb,counter)
    elif key == ord('s'):
        grayandsmooth()
    elif key == ord('S'):
        imagen_new = gaussianfilter()
    elif key == ord('d'):
        imagen_new = downsample()
    elif key == ord('D'):
        imagen_new = resize()
    elif key == ord('x'):
        imagen_new = gradientx()
    elif key == ord('y'):
        imagen_new = gradienty()
    elif key == ord('m'):
        imagen_new = magnitud()
    elif key == ord('p'):
        vectorfield()
    elif key == ord('r'):
        rotate()
    elif key == ord('h'):
        print(help)
    elif key==27:
        cv2.destroyAllWindows()
        break
```

4. 'W': The program have to save the image with the changes in a file called *"out.jpg"*. The program have the current image, with the most recent transformation carried out and use the *.imwrite* function of the cv2 library to save it.

```
def writing(imagen):
    cv2.imwrite("out.jpg",imagen)
```

5. 'g': The program have to convert the image to a grayscale using the openCV conversion function. To perform this, the program will use .cvtColor() and tell to perform a colour to grey change with *"cv2.COLOR_BGR2GRAY"*.

```
def converting(image):
    imagen_new = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    cv2.imshow('CS_512',imagen_new)
    return imagen_new
```

6. 'G': The program will scroll all the pixels of the image and multiply each channel by the change it need to make the switch to black and white image.

$$Channel[0] = Value * 0.299$$
$$Channel[1] = Value * 0.587$$
$$Channel[2] = Value * 0.114$$

```
def convertinggray(image):
    imagen_new = np.zeros((image.shape[0], image.shape[1]), dtype = image.dtype)
    for (x, y), v in np.ndenumerate(imagen_new):
        value = image[x, y, 0] * 0.299 + image[x, y, 1] * 0.587 +
        value += image[x ,y, 2] * 0.114
        imagen_new[x, y] = value
    cv2.imshow('CS_512',imagen_new)
    return imagen_new
```

7. 'c': The program will switch off the other channel depending with one we want. For example, if we want the channel blue: the program will switch off the green and red channels. The program uses a counter to change between the different channels, saving the value and also the channel that has been performed before.

```
def cycle(nb,counter):
    imagen_new = reload(nb)
    if (counter == 0):
        imagen_new[:,:,1] = 0
        imagen_new[:,:,2] = 0
        counter += 1
    elif (counter == 1):
        imagen_new[:,:,0] = 0
```

```
        imagen_new[:,:,2] = 0
        counter += 1
    elif (counter == 2):
        imagen_new[:,:,0] = 0
        imagen_new[:,:,1] = 0
        counter = 0
    cv2.imshow('CS_512', imagen_new)
    return imagen_new, counter
```

8. 'D': The program uses the *.pyrDown()* function to resize the image. The function performs the downsampling step of the Gaussian pyramid construction.

```
def downsample():
    imagen_new = cv2.pyrDown(reload(nb))
    cv2.imshow('CS_512', imagen_new)
    return imagen_new
```

9. 'd': The program uses the function .resize() for perform this transformation. It uses bicubic interpolation, which realizes the resize over 4x4 pixel neighborhood.

```
def resize():
    imagen_new = cv2.resize(reload(nb),None,fx=0.5, fy=0.5, interpolation =
    cv2.INTER_CUBIC)
    cv2.imshow('CS_512', imagen_new)
    return imagen_new
```

10. 's': First, the program changes the color of the image to black and white with the function of openCV. As the image must has a TrackBar, the operations to increase the smooth are performed in the sliderHandler function. In the sliderHandler, the program will apply a filter to the image.

```
def sliderHandler(self):
    global imagen_new
    if self != 0:
        n=self
        imagen = cv2.cvtColor(reload(nb), cv2.COLOR_BGR2GRAY)
        kernel = np.ones((n,n),np.float32)/(n*n)
        imagen_new = cv2.filter2D(imagen,-1,kernel)
    cv2.imshow('CS_512_1',imagen_new)


def grayandsmooth():
    imagen_new = cv2.cvtColor(reload(nb), cv2.COLOR_BGR2GRAY)
    cv2.imshow('CS_512_1', imagen_new)
    cv2.createTrackbar('S','CS_512_1', 0, 255, sliderHandler)
```

11. 'S': Like the example above, but the program applies a Gaussian filter to the image so that it has the final result expected.

```
def sliderHandler_2(self):
    global imagen_new
    if self != 0:
        sigma = 7
        size = self
        result = cv2.cvtColor(reload(nb), cv2.COLOR_BGR2GRAY)
        interval = (2*sigma+1.)/(size)
        x = np.linspace(-sigma-interval/2., sigma+interval/2., size+1)
        kern1d = np.diff(st.norm.cdf(x))
        kernel_raw = np.sqrt(np.outer(kern1d, kern1d))
        kernel = kernel_raw/kernel_raw.sum()
        result = cv2.filter2D(result, -1, kernel)
        imagen_new = result
    cv2.imshow('CS_512_1',imagen_new)

def gaussianfilter():
    imagen_new =cv2.cvtColor(reload(nb), cv2.COLOR_BGR2GRAY)
    cv2.imshow('CS_512_1', imagen_new)
    cv2.createTrackbar('S','CS_512_1', 0, 255, sliderHandler_2)
    result = sliderHandler_2
    return result
```

12. 'x':The program changes the color of the image and implement a Sobel filter to obtain the gradient of x. To perform this, the parameters that we have to put in the function are (x = 1, y = 0). Finally, the program normalizes the result between the values [0,255], but in the function of openCV this parameters are alpha = 0 and beta = 1.

```
def gradientx():
    imagen_new = cv2.cvtColor(reload(nb), cv2.COLOR_BGR2GRAY)
    imagen_new = cv2.Sobel(imagen_new,cv2.CV_64F, 1, 0,ksize = 7)
    imagen_new = cv2.normalize(imagen_new,imagen_new,alpha = 0, beta = 1,
    norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
    cv2.imshow('CS_512', imagen_new)
```

13. 'y': The program changes the color of the image and implement a Sobel filter to obtain the gradient of x. To perform this, the parameters that we have to put in the function are (x = 0, y = 1). Finally, the program normalizes the result between the values [0,255], but in the function of openCV this parameters are alpha = 0 and beta = 1.

```
def gradienty():
    imagen = cv2.cvtColor(reload(nb), cv2.COLOR_BGR2GRAY)
    imagen_new = cv2.Sobel(imagen,cv2.CV_64F, 0, 1,ksize = 7)
    imagen_new = cv2.normalize(imagen_new, imagen_new, alpha = 0, beta = 1,
```

```
    norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
    cv2.imshow('CS_512', imagen_new)
```

14. 'm': The program performs the gradient of x and the gradient of y. The function magnitude calculates the magnitude of 2D vectors formed from the corresponding elements of x and y arrays

$$dist(I) = \sqrt{x(I)^2 + y(I)^2}$$

Finally, the program normalizes the result between the values [0,255], but in the function of openCV this parameters are alpha = 0 and beta = 1.

```
def magnitud():
    sobelX = cv2.Sobel(reload(nb),cv2.CV_64F, 1, 0,ksize = 7)
    sobelY = cv2.Sobel(reload(nb),cv2.CV_64F, 0, 1,ksize = 7)
    imagen = cv2.magnitude(sobelX, sobelY)
    imagen = cv2.normalize(imagen,imagen, alpha = 0, beta = 1,
    norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
    cv2.imshow('CS_512', imagen)
    return imagen
```

15. 'r': The program rotates the image and then applies an affine transformation to the image.

```
def sliderHandler_3(self):
    global imagen_new
    n=self
    imagen = cv2.cvtColor(reload(nb), cv2.COLOR_BGR2GRAY)
    rot = cv2.getRotationMatrix2D((imagen.shape[1] / 2,
    imagen.shape[0] / 2), n, 1)
    imagen = cv2.warpAffine(imagen, rot,(imagen.shape[1], imagen.shape[0]))
    imagen_new = imagen
    cv2.imshow('CS_512_1', imagen)

def rotate():
    cv2.imshow('CS_512_1', reload(nb))
    cv2.createTrackbar('S','CS_512_1', 0, 360, sliderHandler_3)
```

16. 'p': The program performs the gradient of x and the gradient of y, and with both values describes the angle of the gradient. Then, the program obtains the cos and the sin of the gradient so it could draw the line. The function arrowedLine draws an arrow between pt1 and pt2 points in the image.

```
def sliderHandler_4(self):
    global imagen_new
    if self != 0:
```

```
        n=self
        img = cv2.cvtColor(reload(nb), cv2.COLOR_BGR2GRAY)
        sobelX = cv2.Sobel(img,cv2.CV_64F, 0, 1,ksize = 7)
        sobelY = cv2.Sobel(img,cv2.CV_64F, 1, 0, ksize = 7)
        for x in range(0, img.shape[0], n):
            for y in range(0, img.shape[1], n):
                gradientAngle = math.atan2(sobelY[x, y], sobelX[x, y])
                dstX = int(x + n * math.cos(gradientAngle))
                dstY = int(y + n * math.sin(gradientAngle))
                cv2.arrowedLine(img, (y, x), (dstY, dstX), (0, 0, 0))
    imagen_new = img
    cv2.imshow('CS_512_1', img)

def vectorfield():
    cv2.imshow('CS_512_1', reload(nb))
    cv2.createTrackbar('S','CS_512_1', 0, 255, sliderHandler_4)
```

## 3. Implementation details

- **Problems found during the implementation of the program:**

**Problem 1** : The trackbar did not disappear from the main window in case the image did not need it. To solve it, I have implemented two windows: one with and another without a trackbar.

**Problem 2** : In the cycle function, I could not save the previous value if the function was changed to another and then return to the cycle one again. So at the end, I save this value in a variable *counter* and return it to the main function to save the result without needing global variables.

**Problem 3** : At first, when I was using the trackbar I was performing the operations to the previous image and not to the original. Therefore, the results were overlapping one to each other. I had to load the original image every time the function was executed.

- **Instructions for using the program:**

**Step 1** : Write a file name of a image in the same foulder as this or press Enter.

**Step 2** : If you put a file name, you should see the image in the screen. Otherwise, you will be taken a photo with the camera of your laptop, please smile!

**Step 3** : Press any of this bottom in the keyboard of your laptop: i, g, G, c, s, S, d, D, x, y, m, p, r or h.

**Step 4** : To finish press the Esc button.

## 4. Results and discussion

- **Discuss of the correctness of the implementation:**

1. Upload an image from the computer: The program can upload a file from the computer. The file must be in the same folder as the program, the user can not put the path. Also if the user write the wrong name, the program will notice that and the user to write the file name again.

2. Use an image from a camera: The program is capable of capturing an image from a camera. In our case, it is using the camera of the computer.

3. The images obtained are loaded with three channels.

4. The program can switch between different states using the original image or a new one. Therefore, it does not make the changes on the previous changes made in the image.

5. 'i': The program is able to reload the previous image: uploading the original or capturing a new image.

6. 'w': The program is able to create a .jpg image file by saving the last transformation made during the program execution.

7. 'g': The program is able to change the color to a black and white image.

8. 'G': The program is able to change a black and white image, with different results than in the previous case.

9. 'c': The program is able to leave only one channel in the image and visualize it. In addition, it perform a cycle between the different channels, saving the state. So, you can press another key and if you come back, the program will perform the next channel. It will not start again the cycle.

10. 's': The program is able to smooth the image depending on the value of the trackbar that appears in the top of the image. Consequently, I can see how the smooth value increases with different values in trackbar. This result appears in a new window therefore the user can see the trackbar. when I putted the trackbar in the same window as the last one, the trackbar was left the window and I wasn't able to remove it.

11. 'D': The program is able to reduce the size of the image by a factor 2 without smooth.

12. 'd': The program is able to reduce the image size with smooth, using a Gaussian pyramids.

13. 'x': The program is able to obtain the gradient of x in the image. I perform the result with the same image and see that there are different between the gradient x and the gradient y. In one of them the highlighted lines are in one direction and in the other in perpendicular direction.

14. 'y': The program is able to get the gradient of y in the image.

15. 'm': The program is able to obtain the sum of the two gradients x and y, the magnitude of the image.

16. 'p': The program is able to obtain the vectors of the image depending on a trackbar and the value of each gradient.

17. 'r': The program is able to rotate the image among 360 degrees.

- **Result obtained:**


Figure 1: The result obtained with the option 'g'.


Figure 2: The result obtained with the option 'G'.


Figure 3: The result obtained with the option 'c'. Channel 0.


Figure 4: The result obtained with the option 'c'. Channel 1.


Figure 5: The result obtained with the option 'c'. Channel 2.


Figure 6: The result obtained with the option 's'.Trackbar = 50

Figure 7: The result obtained with the option 'S'. Trackbar = 150


Figure 8: The result obtained with the option 'D'.


Figure 9: The result obtained with the option 'd'.

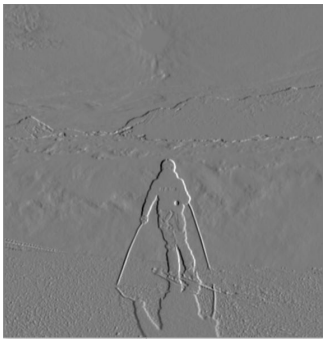

Figure 10: The result obtained with the option 'x'.

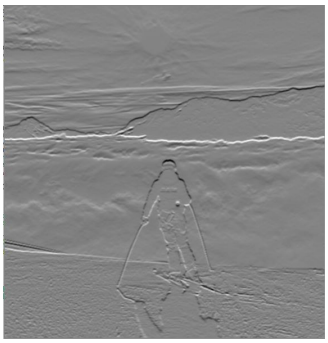

Figure 11: The result obtained with the option 'y'.


Figure 12: The result obtained with the option 'm'.


Figure 13: The result obtained with the option 'p'. Trackbar = 20


Figure 14: The result obtained with the option 'p'. Trackbar = 80


Figure 15: The result obtained with the option 'r'. Trackbar = 70


Figure 16: The result obtained with the option 'r'. Trackbar = 150