LABORATORY 9: Embedded computer vision II

Part 1: Image filtering.

Theory Concepts:

Blur filters: These filters allow the users to eliminate image noise and improve the target object presentation to enhance future operations like border detection, object counting, etc.

Some of the most important blur filters we can use are:

- Averaging blur
- Median blur
- Gaussian blur

Averaging blur:

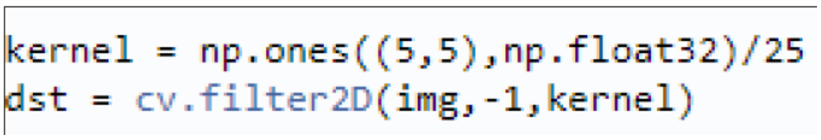
We will use an odd positive kernel to modify our image.

$$K = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

To ease the kernel generation in python we can use the numpy library with the **ones** function.

- **What are ones in numpy, how does it work?**

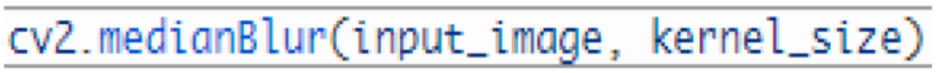
The following code lines are an example of how to use the averaging blur:



```
kernel = np.ones((5,5),np.float32)/25
dst = cv.filter2D(img,-1,kernel)
```

Median blur:

The median blur will use the median value of a matrix to replace the center value of that matrix. To use it we must define the image and the kernel size:

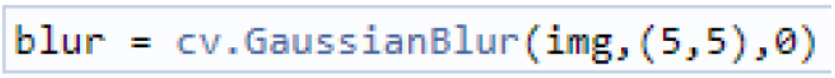


```
cv2.medianBlur(input_image, kernel_size)
```

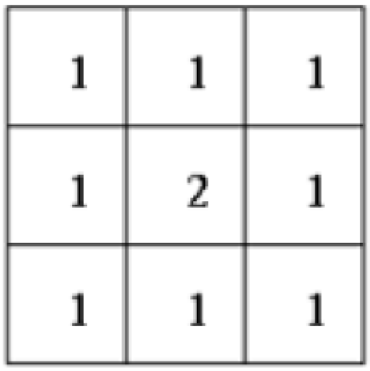
- **How do we find the median of a matrix?**

Gaussian blur:

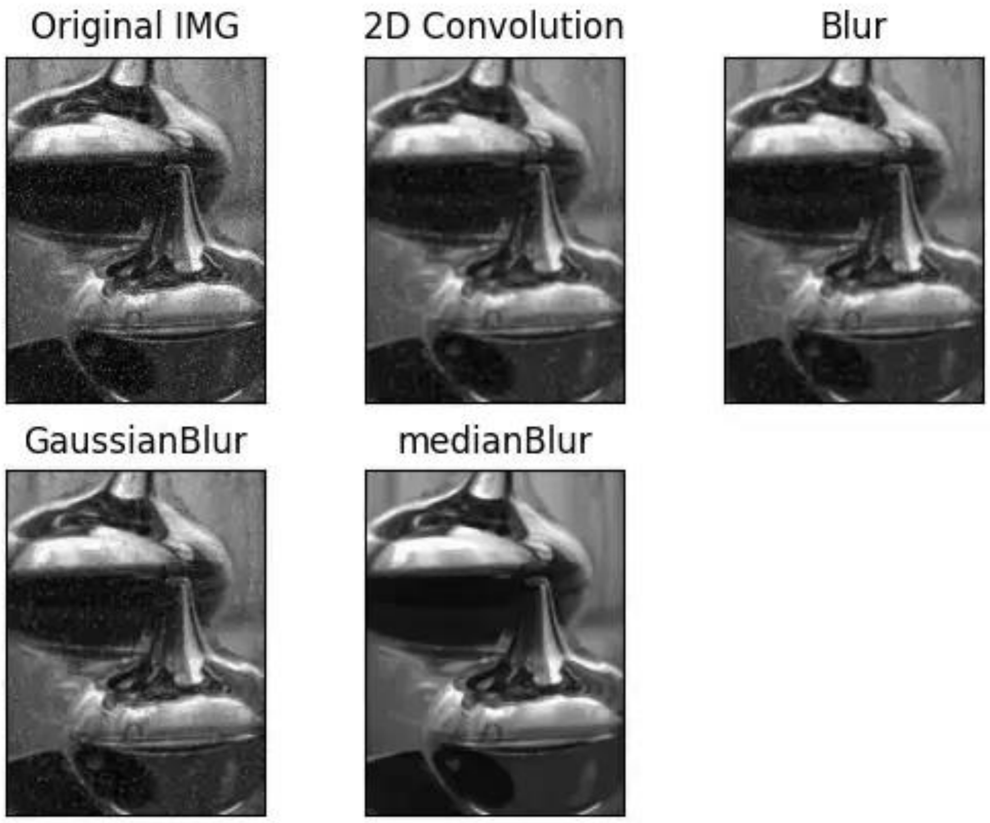
One of the most common used filters is the Gaussian blur, this one focusses the image transformation on the center pixel of the specified matrix. To use Gaussian blur, we define the following OpenCV function:



```
blur = cv.GaussianBlur(img,(5,5),0)
```



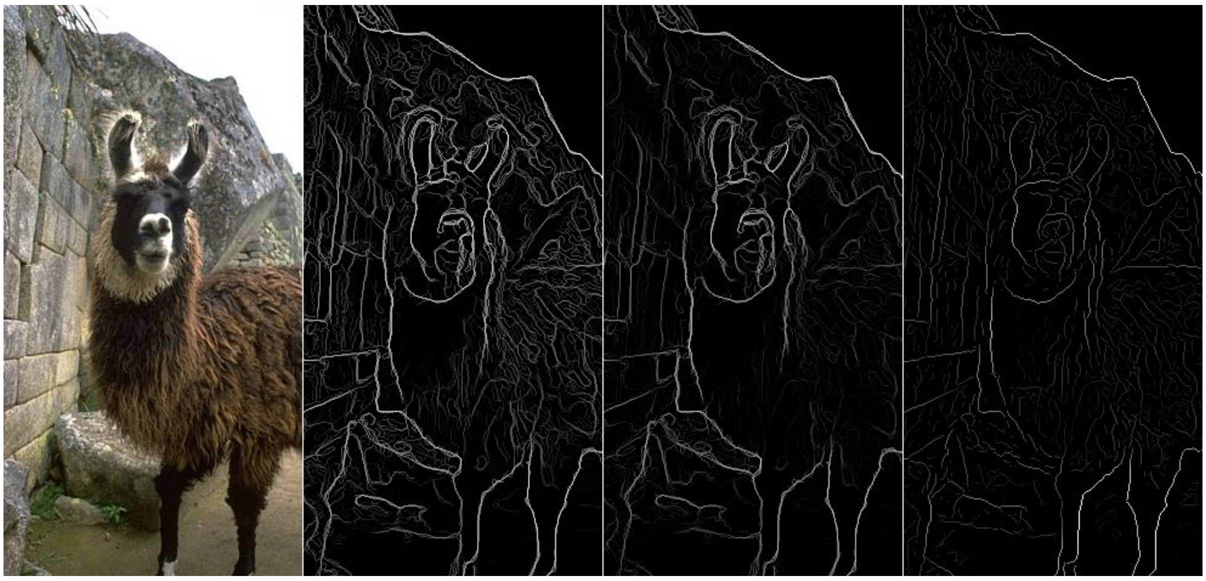
1	1	1
1	2	1
1	1	1



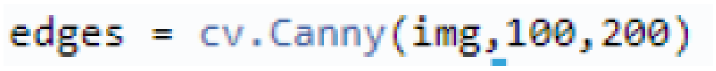
Edge detection:

To detect edges, we can implement high pass filters that will allow to pass the pixels that generate high contrast values like borders. One of the most used filters is the Canny edge detection filter.

This filter has a special feature that will use minimum and maximum threshold values, as we can see the example below, we can modify the threshold values and increase or reduce the edges detected by this filter:



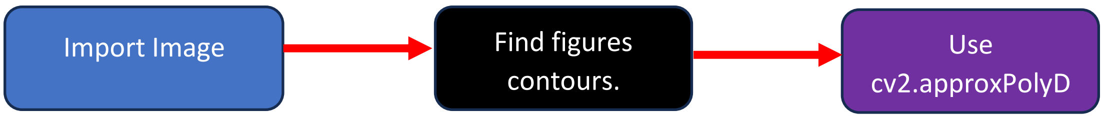
To use Canny, we can implement the following code line:



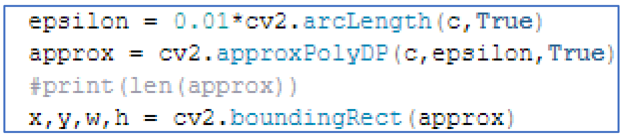
```
edges = cv.Canny(img,100,200)
```

Figure detection with approxPolyDP:

We can use some OpenCV specific functions to detect geometric figures, for that we need to:



The approxPolyD function will use the image with the contours, an epsilon value that helps the approximation and a “closed” parameter that will indicate if the figure is a closed shape.



```
epsilon = 0.01*cv2.arcLength(c,True)
approx = cv2.approxPolyDP(c,epsilon,True)
#print(len(approx))
x,y,w,h = cv2.boundingRect(approx)
```

- **What does cv2.acrLenght do?**
- **How can you differentiate a square from a rectangle?**

To know which figure we are receiving we will use the length of the “approx” variable as it contains the number of lines the figure has.

Exercises:

1. **Remove the background from the “bouncing” mp4 file. To do that you need to test the subtractors in the cv2 library and see the differences between them. After the comparation choose one of the subtractors for the next exercise.**
2. **Remove the background from the webcam and test it moving an object. To do that, the webcam must be steady and pointing to a fix background. Finally, when the object goes through the middle of the screen the message “Object Detected” must appear in the bottom right of the frame.**
3. **For the “bouncing” mp4 file find and draw the contours of the objects that are in movement.**
4. **From the image “monedas.jpg” generate a program that can detect the number of coins in the image.**
5. **Using the results from exercise 2 combine TIVA and Raspberry to make an early-stage object tracker robot. For that the system must use 2 motors and wheels to move in the direction of the object detected.**
6. **How would you make the object to find objects? Explain your answer.**
7. **Using exercise 5 modify the system to:**
    a. **Follow a single object.**
    b. **If there are 2 or more objects toggle 2 LEDs.**
    c. **If there is no object make the system to turn around slowly.**
8. **Using the “figuras” image detect the type of geometric figure and its color and show the results in a new frame.**