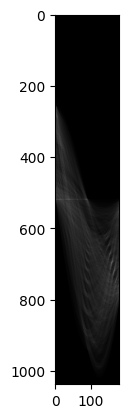
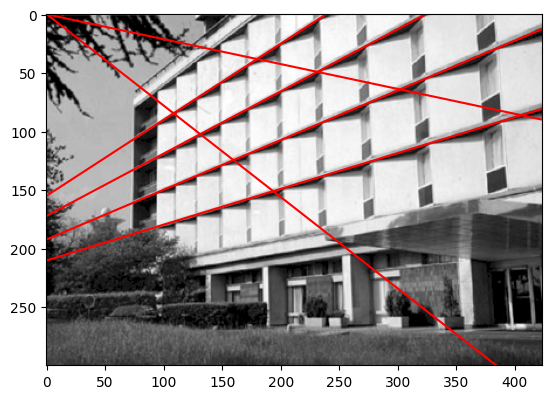
# **Lines detection using Hough transform:**

This algorithm detects lines by finding the (ρ, θ) pairs that have a number of intersections larger than a certain threshold, we can adjust the threshold and notice the difference as we will see later, here, we’ve used the built-in function of canny edge detection algorithm of open-cv2 library, as we find its results more accurate than the one we built from scratch so we preferred the built-in one

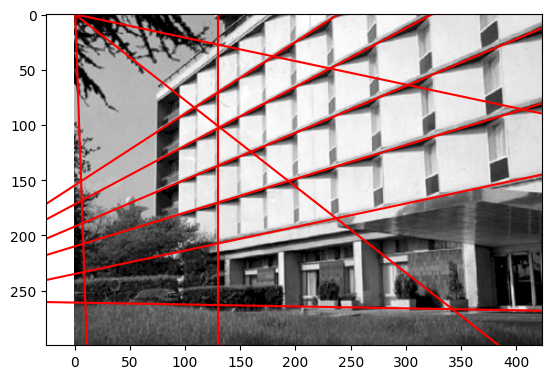
Now, we will present some of the results of this method;

* Here we have the original image:
* Here is the image in Hough transform:
* And finally, here are the detected lines:



In the previous image, I set threshold1 &2 in Canny’s to 150 &150.

* Let’s change it to 100&150 and notice the difference, the resulted lines:



So, new 3 lines appeared when we change the threshold, so we can control the resulted lines by adjusting the threshold.

# **2nd, Hough transform for circles:**

* The source of the code for this part: https://www.codingame.com/playgrounds/38470/how-to-detect-circles-in-images

In this part, we are doing canny edge detection manually instead of using the built-in function, we set the Rmin &Rmax (max and min radiuses range we want to detect) and threshold, the algorithm here usually performs well at thresholds of 0.4 or 0.2.

Let’s see some examples;

* Here is the photo I’m going to apply my function on:



* The result is:



As we can see, it detected the circles accurately, if we change the threshold, would it matter? Let’s see

In the previous image, the threshold was set to 0.4, let me change it to 0.6 instead and see how that would affect our detection;

* As we see, an unsuitable threshold would create other circles that don’t really exist, so make sure you’ve used a suitable threshold for your image.
* Also, if the range of radiuses you set (Rmin to Rmax) doesn’t include the radiuses of circles in the image, the algorithm wouldn’t even see those circles (kinda makes sense).
* By setting the ranges smaller values than those of the coins’ radiuses, the algorithm has gone crazy(wasn’t really expected):



* Also, one disadvantage of this method, is that if there were many votes in a specific place, with not really accurate threshold, it’s going to mess up and detect not existent circles.

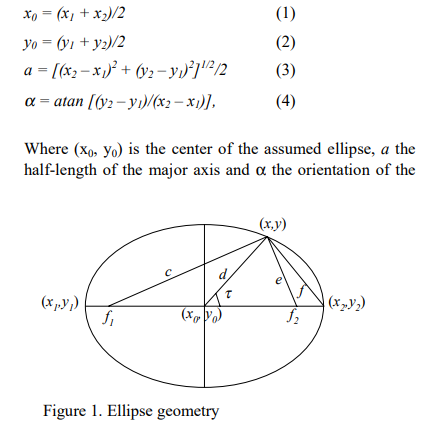
****

* So if you want to detect circles more accurately choose:

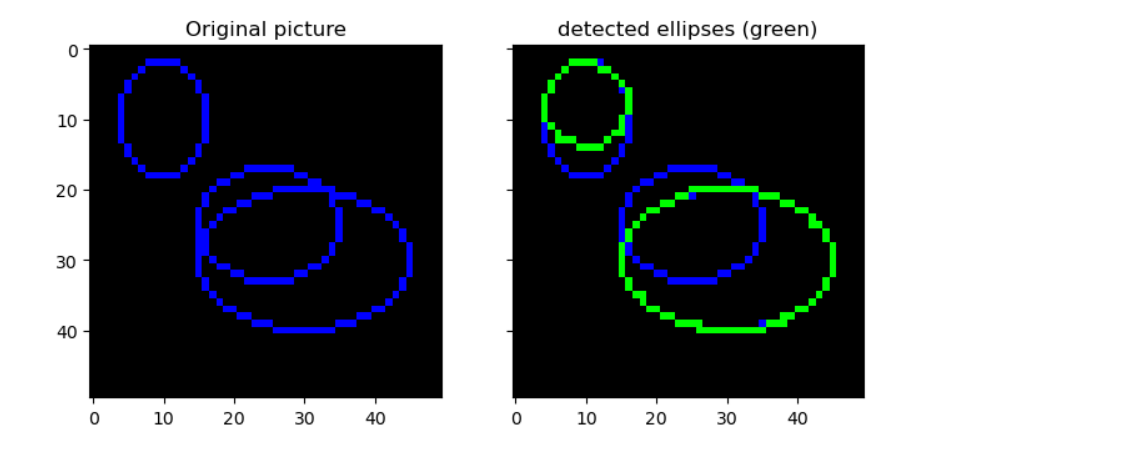
1. Suitable threshold (I think we can know this one by trial and error).
2. Suitable range for rmin to rmax.

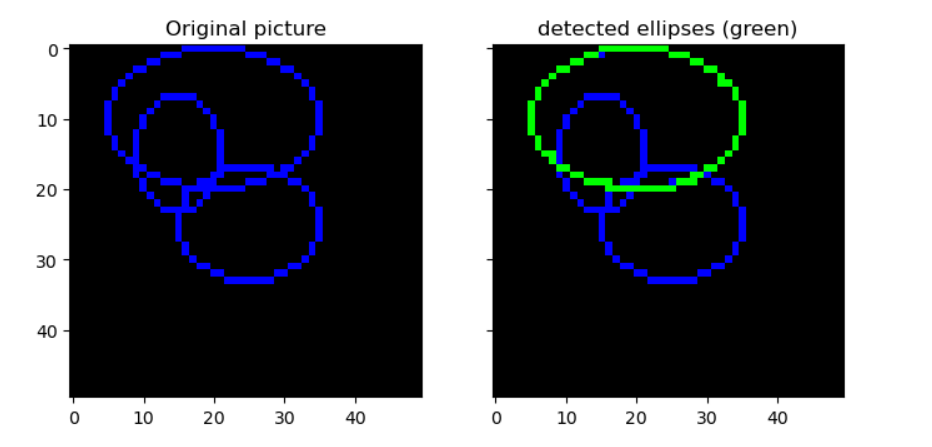
# **Ellipse detection algorithm:**

* We have implemented the [ellipse detection algorithm](https://sites.ecse.rpi.edu/~qji/Papers/ellipse_det_icpr02.pdf) published by Xie, Yonghong, and Qiang Ji. The algorithm finds the 5 parameters for each ellipse that could be in the original image and a voting threshold is applied to the accumulator to return the best possible shapes. For each pair of pixels (x1, y1) and (x2, y2), we assume they are two vertices on the major axis of an ellipse and the parameters are calculated from the equations found in the paper from ellipse geometry.



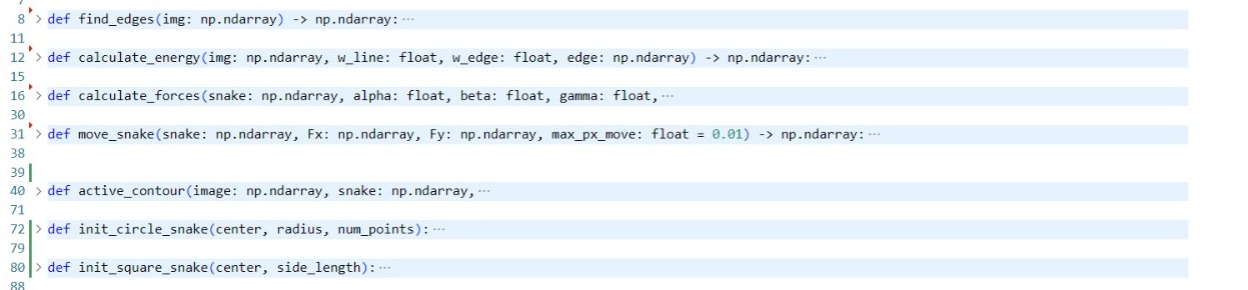
So, the algorithm is basically as follows after constructing the pixels array from the edges of the image:

* For each pixel we find another pixel and assume they are the points for the major axes (if is greater than min length in function) and then we calculate the ellipse parameters from its geometry and the equations above, then for each third pixel we calculate the length of the minor axis and increment the accumulator of the found ellipse by one, if the accumulator of an ellipse votes exceed a certain threshold then we add it to out result.
* The algorithm does well mostly but is very computationally expensive so it takes a long time for larger photos and the parameters needs to be supplied correctly because it is very sensitive to changes in threshold and the accuracy( the parameter used to get the ellipse from the accumulator distribution for each third pixel).
* Below is our function output for some ellipses synthetic images as we found it does better for images with less background noise: 



* Here we have plotted some of the detected ellipses result against the original image, we do not plot all the results because the highest votes can indicate the same ellipse more than once but with a small shift.

# **Contouring:**



In this part, we will discuss the contouring process

* The active contour function uses the Sobel operator to find edges in the image, and then calculates the internal and external forces acting on the snake using the energy function, alpha, beta, and gamma parameters. The snake is then moved based on these forces until convergence is achieved or the maximum number of iterations is reached.
* The RectBivariateSpline function is used to create a spline interpolation of the image intensities, and the move\_snake function updates the snake's position based on the forces acting on it.
* We have 2 functions to initiate the contour, the user can start with a circle or a square snake or can initiate the snake points one by one.
* The user also can generate a blank image to test the function or browse an image.
* One possible improvement to consider is the use of the chain code, the chain code is a compact representation of a contour that encodes the direction of successive points along the contour as a series of integers. By using it, the algorithm can better capture the shape of the contour and reduce the number of points needed to represent it accurately. This, in turn, can lead to faster convergence and better results.
* Incorporating the chain code into the active contour code may require some modifications to the existing functions, particularly those that deal with the calculation of internal and external forces. Additionally, the initialization of the snake may need to be adjusted to use the chain code instead of a set of points. However, these changes can be made relatively easily and can lead to a more efficient and accurate implementation of the active contour algorithm.
* The values of alpha, beta, and gamma need to be carefully chosen to balance the competing effects of internal and external energies and to achieve the desired behavior of the snake,

1. Alpha: controls the rigidity of the snake.
2. Beta: controls the elasticity of the snake.
3. Gamma: controls the influence of the external energy term.

* These are 2 contours generated using different parameters :

