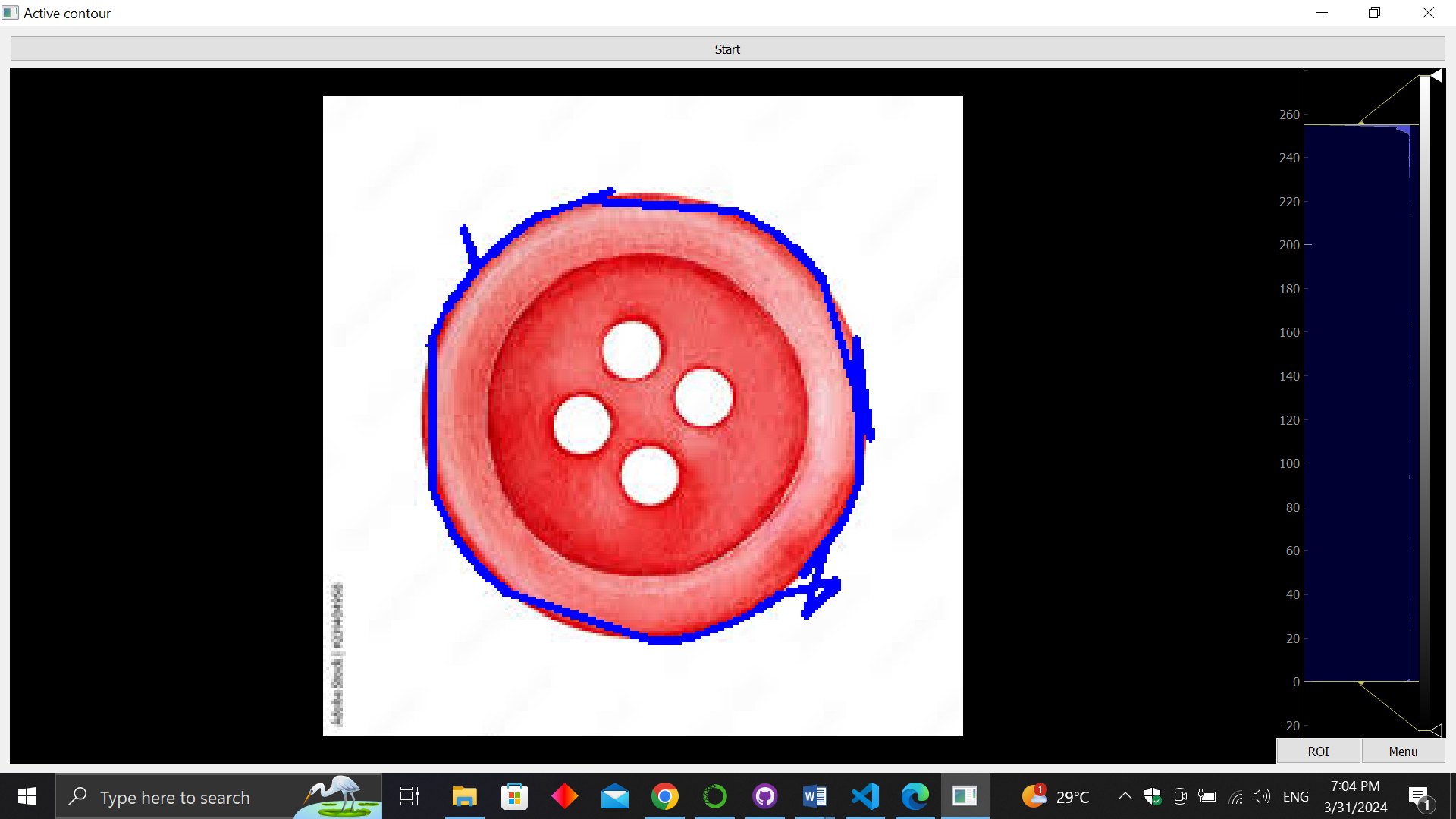
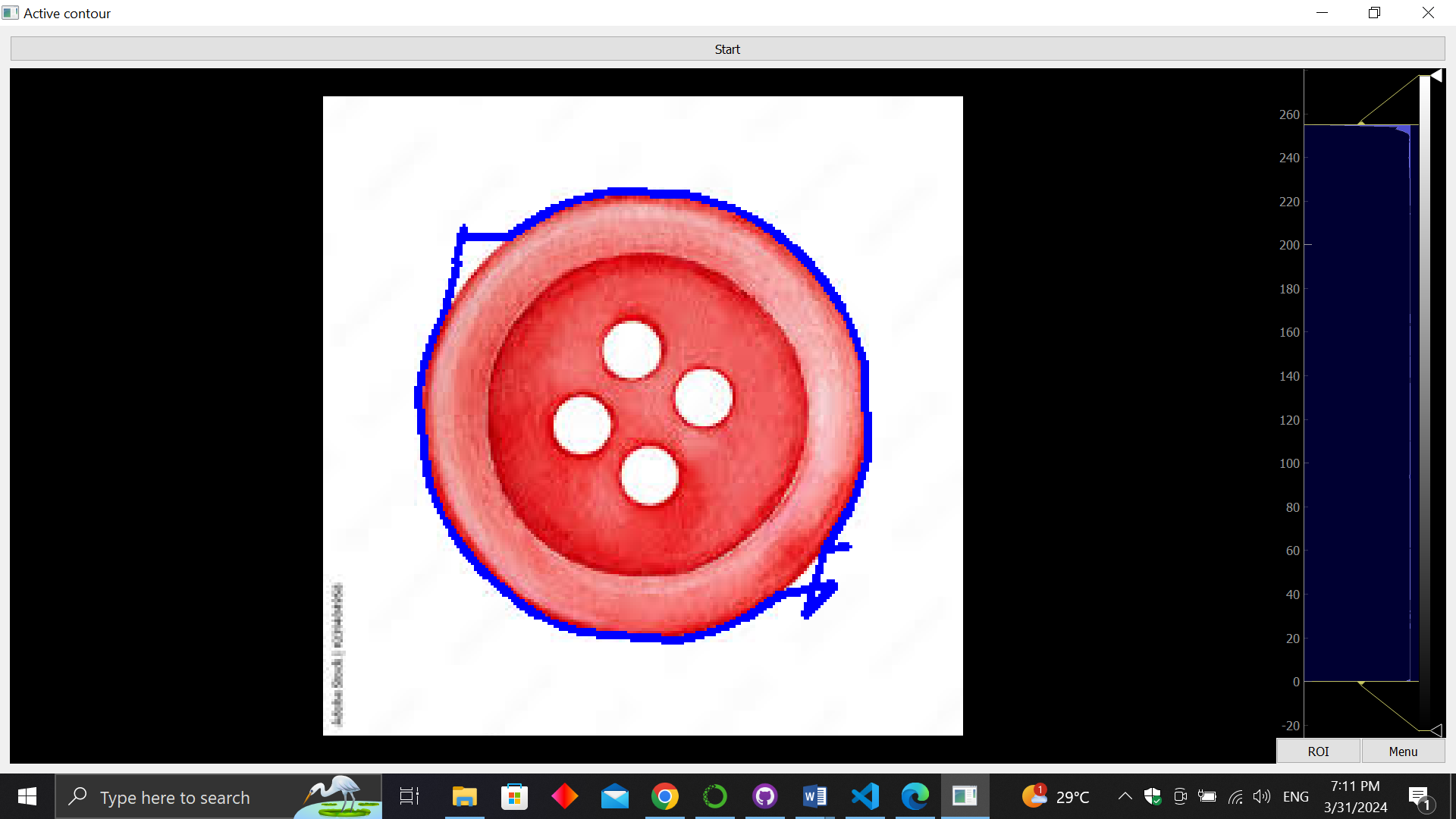
**Active Contour Model (Snake Algorithm)**

1. **create\_initial\_contour function**:
   * **Parameters**:
     + **source**: Input image as a numpy array.
   * **Description**:
     + Generates an initial contour for active contour models.
     + Uses a circular contour with a specified number of points.
     + The center of the circle is the center of the image, and the radius is 40% of the minimum dimension of the image.
2. **internal\_energy function**:
   * **Parameters**:
     + **contour\_points**: Array of shape (N, 2) containing the coordinates of N points defining the initial contour.
     + **alpha**: Weight parameter controlling the contribution of elasticity to the internal energy.
     + **beta**: Weight parameter controlling the contribution of stiffness (curvature) to the internal energy.
   * **Description**:
     + Calculates the internal energy of a contour defined by a set of points using the following formula:
     + Consists of elasticity and stiffness (curvature) components.
     + Elasticity (first term) is based on the difference between distances of consecutive points from the mean distance.
     + Stiffness (second term) is based on the curvature of the contour.
   * **Observations**:
     + Balancing **alpha** and **beta** is crucial for proper contour convergence. Higher values of **alpha** prioritize elasticity, while higher values of **beta** prioritize contour smoothness.
3. **prepare\_external\_energy function**:
   * **Parameters**:
     + **image**: Input image as a numpy array.
   * **Description**:
     + Prepares external energy components for active contours (intensity and edges).
     + Applies Gaussian blur to the input image to reduce noise.
     + Computes edges/edge map using the Sobel filter to highlight object boundaries.
   * **Observations**:
     + Blurring helps in smoothing the image, which can improve the accuracy of edge detection.
     + Sobel edge detection emphasizes **gradient** changes, which are indicative of object boundaries.
4. **external\_energy function**:
   * **Parameters**:
     + **point**: Coordinates of the point (x, y).
     + **intensity**: Intensity values of the image.
     + **gradient**: Gradient magnitude of the image.
     + **w\_line**: Weight for intensity term.
     + **w\_edge**: Weight for gradient term.
     + **gamma**: Scaling factor.
   * **Description**:
     + Gets external energy at a given point based on intensity and gradient information.
     + Combines intensity and gradient energies with specified weights.
   * **Observations**:
     + **gamma** adjusts the overall influence of external energy on the contour evolution.
     + Proper selection of **w\_line** and **w\_edge** balances the contribution of intensity and gradient information to the contour evolution.
5. **perform\_active\_contour function (main process)**:

* **Description and Steps:**
  + It iterates over a specified number of iterations (termination variable), updating the contour at each iteration.
  + Within each iteration, it iterates over each point in the contour and explores neighboring locations to minimize the total energy, which is a combination of internal and external energies.
  + It updates the contour based on the point with minimum energy.
  + Drawing the contour on the original image for visualization of the segmentation process in real-time.
  + It repeats the process for the specified number of iterations.
* **Observations:**
  + The weights w\_line and w\_edge determine the influence of intensity and gradient information on the contour evolution, respectively.
  + The performance of the algorithm depends mainly on the proper selection of parameters such as alpha, beta, gamma, and the number of iterations.

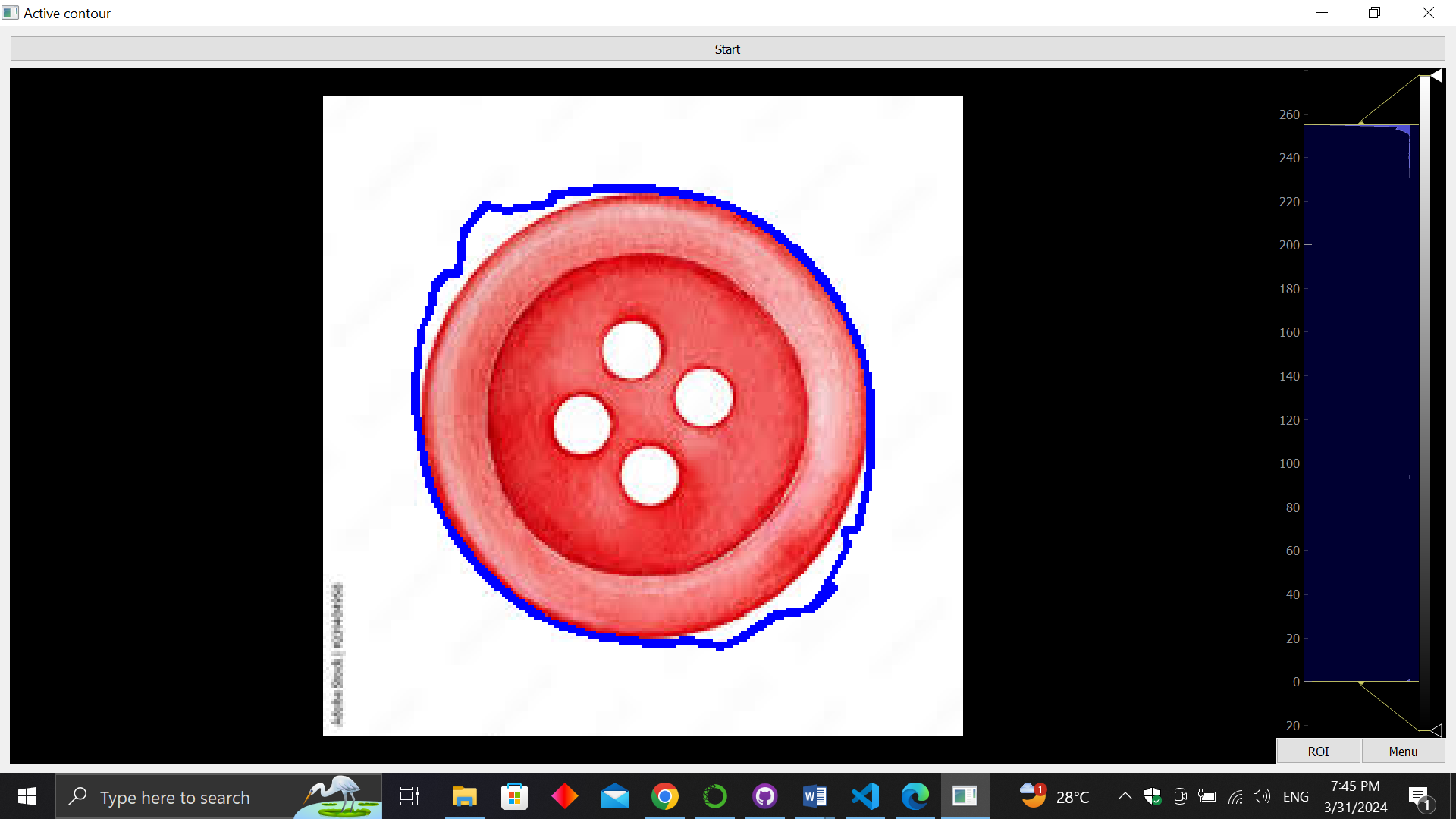
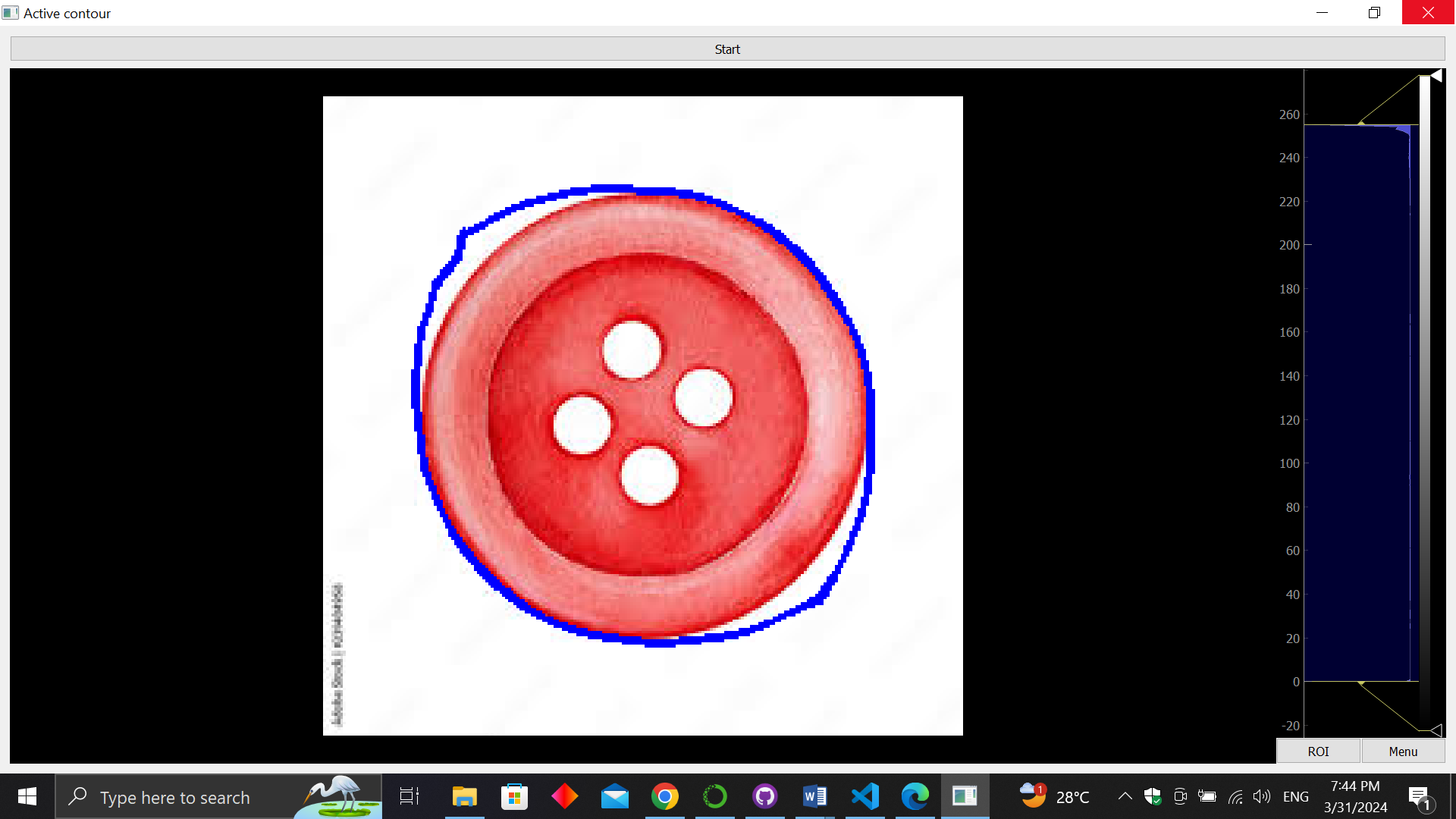
***Experiments:***



For the same alpha and beta values decreasing gamma prioritizes contour shape over image features hence the contour misses the object’s boundaries.

alpha=0.1, beta=1, gamma=5

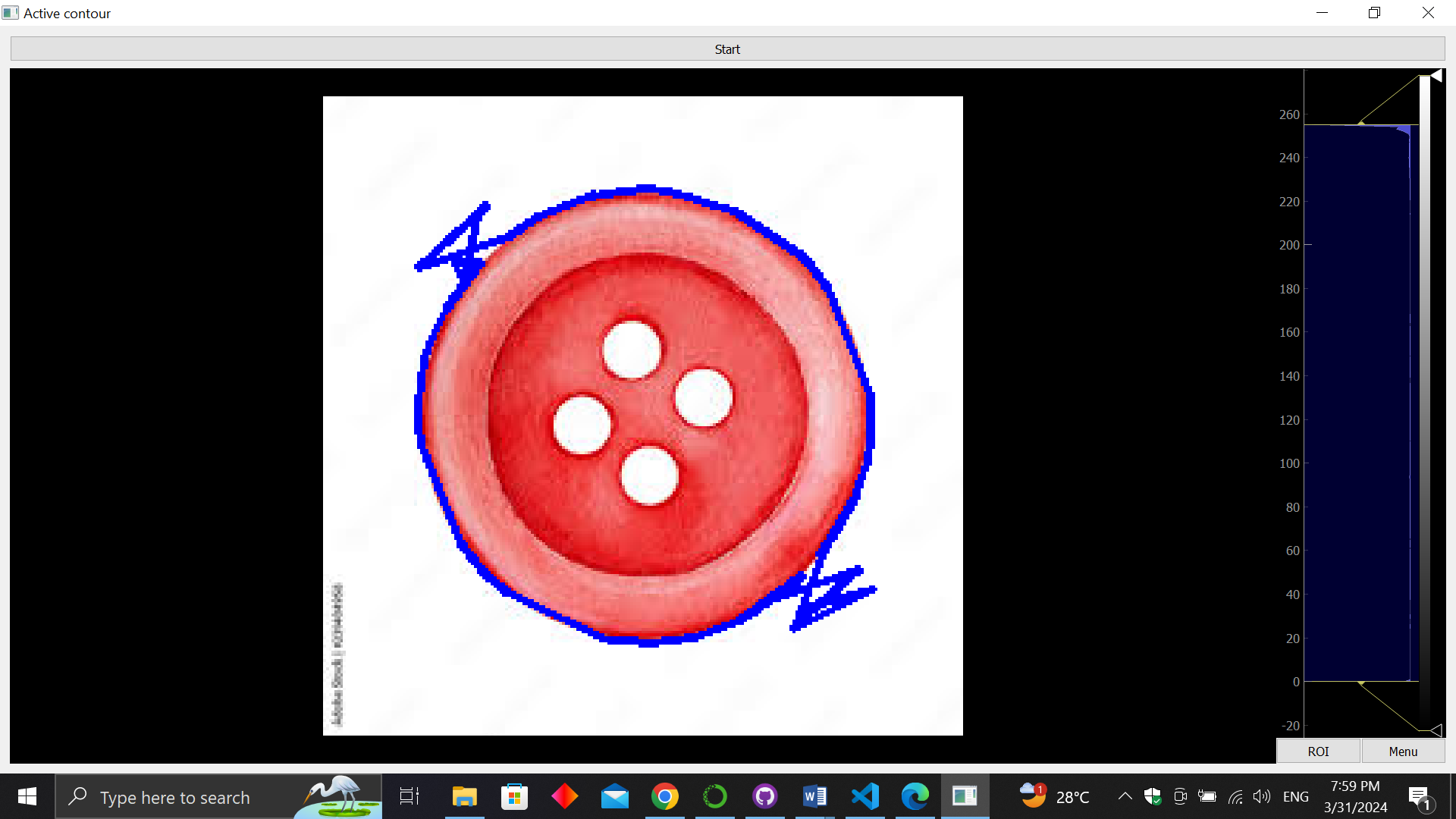
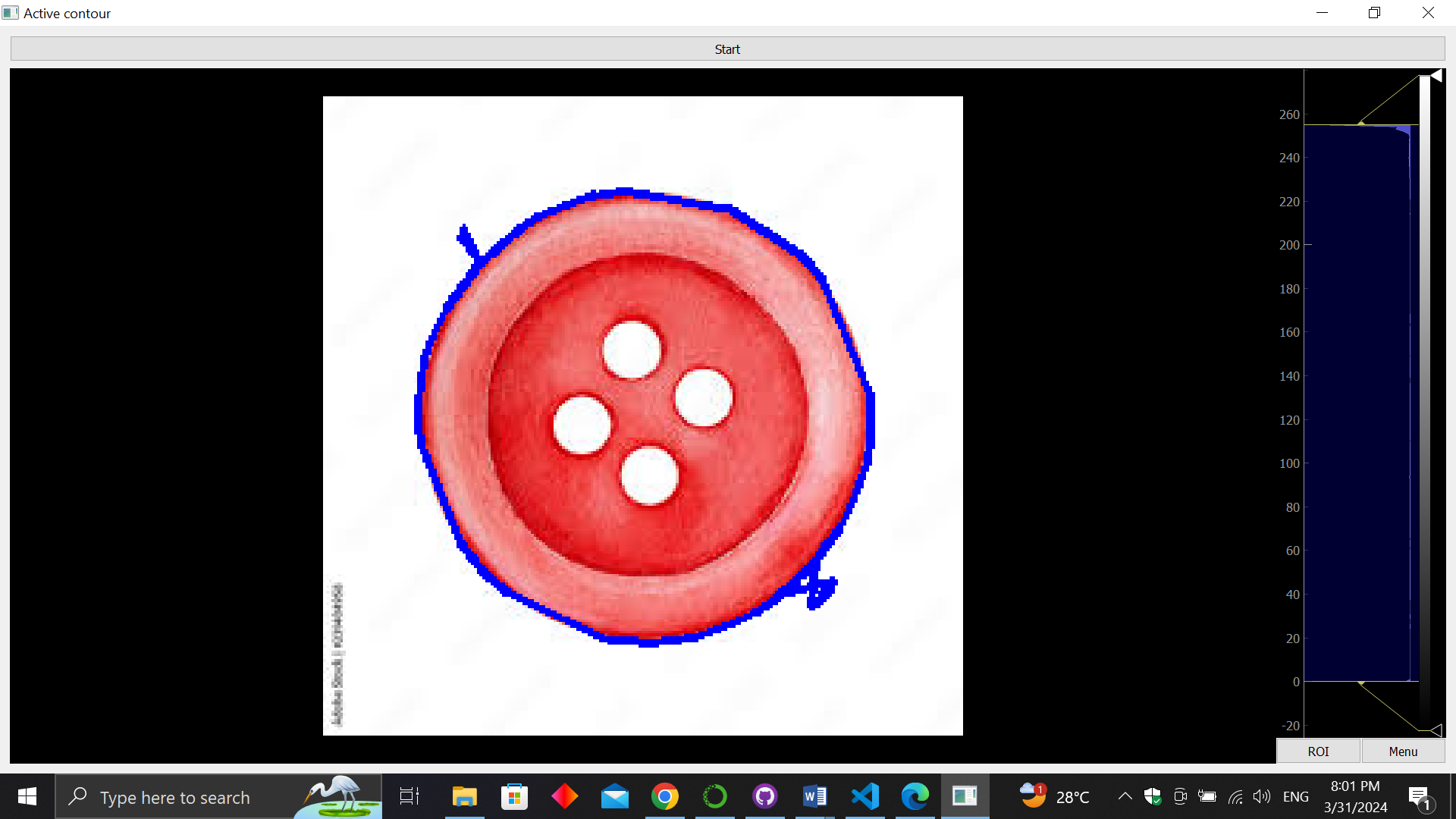
alpha=0.1, beta=1, gamma=2



For the same alpha and gamma values increasing beta makes the contour smoother.

alpha=0.1, beta=1, gamma=10

alpha=0.1, beta=4, gamma=10



alpha=0.1, beta=1, gamma=4

alpha=1, beta=1, gamma=4

For the same beta and gamma values, higher values of alpha result in a stiffer contour, it tends to maintain its shape and resist deformations caused by external forces (image features). Decreasing alpha makes the contour more flexible and allows it to deform more easily.

1. **get\_perimeter function:**

**Description**

    Calculate the perimeter of a polygon given its vertices using Euclidean distance.

**Parameters**

    pts (ndarray): A 2D ndarray of shape (no of points, 2) representing the x

                    and y coordinates of the vertices of the polygon.

**Returns**:

    perimeter (float): The perimeter of the polygon.

1. **get\_area function:**

**Description**

  Calculates the area of a polygon given its vertice.

**Steps**:

* Multiplying each x coordinate by the y coordinate of the next point.
* Multiplying each y coordinate by the x coordinate of the next point
* Summing these 2 products
* Subtracting the sums
* Dividing the result by 2.

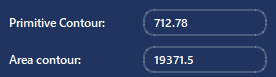
**Parameters**

    pts (ndarray): A 2D ndarray of shape (no of points, 2) representing the x and y coordinates of the vertices of the polygon.

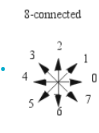
**Returns**:

    area (float): The area of the polygon.

**Sample** **Output**:



# Chain Code

**Those functions collectively produce the chain code for the active contour produced previously, using the 8-connected chain code convention.**

## get\_direction:

**Parameters:**

**curr\_pt** (ndarray): a numpy array containing the x and y coords of the second pt

**prev**\_pt (ndarray): a numpy array containing the x and y coords of the first pt

**Returns**:

**orient** (string): carrying the orientation in one of 8 directions (N, S, E, W, NE, NW, SE, SW)

**Main Point:**

Compares the coordinates of 2 consecutive contour points to decide the orientation (ex: N, SE, NW) of the line segment joining them. Subtracting the x and y coordinates and checking for their signs.

## directions\_to\_chain\_code:

**Parameters:**

**directions** (ndarray): a 2D numpy array produced by the snake algorithm, its shape is (no of points, 2) to produce its chain code

**Returns**:

        chain (list): A list the same size as directions carrying the corrsponding chain codes

**Description:**

Maps every direction (ex: E, W, SN) to its corresponding number in chain code using a list ['E', 'NE', 'N', 'NW', 'W', 'SW', 'S', 'SE'], where each direction has the index of its corresponding number in chain code.

## print\_chain\_code:

**Parameters:**

**pts\_2d** (ndarray): a numpy array containing directions (char)

**Steps:**

* Loops on the whole array, comparing 2 successive points to get the direction of the line segment joining them using the get\_direction function (1), appending all directions to the chain\_directions list.
* Then converting this list of directions (ex: “N”, “SW”) to their corresponding numbers in chain code using the directions\_to\_chain\_code function.
* Turning this list of integers to a string using the list\_to\_string functions which iterates on the list and appends to an empty string.
* Printing into the console with the option to export to a txt file using the export\_chain\_code function.

**Experiment**:



This is the chain code for a circular active contour of a coin, we can see that the abundance of 3, 5, 7, 1 pattern specifies north west then south west then south east and north east making a full circle.