PROJECT DESCRIPTION: (*A brief description of the problem statement. Discuss any relevant assumptions if necessary.*)

1. **Mathematical morphology:** (25 points) Perform an appropriate thresholding on the coins.bmp so that the image contains only black (foreground) and white (background). Then, select an appropriate structure element and appropriate mathematical morphological operators (e.g., closing, opening) to operate on the thresholded coins.bmp such that the coins are well separated from one another. Show the results at each step.

2. **Connected Component Labeling:** (25 points) Apply the recursive connected component labeling on the result of the above mathematical morphology to label the coins. From this, determine the number of coins. Show the image result with different colors or shades of the coins (each coin has a distinct color or shade).

3. **Line Detection:** (25 points) Perform an edge detection on the gray level B2Bomber image. Use an appropriate edge detector. Then, apply the Hough Transform to detect the lines present in the edge detected image

4. **Corner Detection:** (25 points) Apply any corner detector to detect corner points on the gray level B2Bomber. Do not perform edge detection on the image.

PROJECT BACKGROUND: (*Provide a brief summary of background or theory of the techniques used to implement the project.*)

- **Thresholding:** threshold is used to clip a gray-scale image into a binary image. If a pixel's intensity exceeds the threshold, set that pixel to black. If a pixel's intensity succeeds the threshold, set that pixel to white.

- **Dilation:** Dilation expands objects into background. Dilation eliminates "salt" noise. Dilation removes cracks in objects. Use a structural element to dilate an image. Common structural elements are circles and squares.

$$A \oplus B = \cup_{b \in B} A_b$$

- **Erosion:** Erosion shrinks objects. Erosion removes pepper noise. Erosion removes thin object hairs. Use a structural element to erode an image.

$$A \ominus B = \cap_{b \in B} A_{-b}$$

- **Closing:** Closes gaps. Dilation followed by erosion.

$$A \bullet B = (A \oplus B) \ominus B$$

- **Opening:** Opens gaps. Erosion followed by dilation.

$$A \circ B = (A \ominus B) \oplus B$$

- **Object Labeling and Counting (Connected Component Labeling):** the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics. This multi-pass recursive algorithm is used to identify connected pixels; a set of connected pixels represents one coin.

- **Edge Detection (Laplacian of Gaussian (LoG)):** The LoG is composed of two parts; first the Gaussian, second the Laplacian. Applying the Gaussian to an image will smooth the image. The Gaussian has a tendency to blur images. This means that for impulse noises with spikes, the Gaussian will smear spikes over a sizable number of pixels (Gaussian smoothing). The Laplacian highlights regions of rapid intensity change and is therefore often used for edge detection. The Laplacian of an image is a measure of the sum of the second partial derivatives. Edges are highly correlated to the partial second derivatives of the Gaussian.

Zero-crossing the LoG is an edge detection method. A pixel crosses-zero if it is significantly darker than the other pixels in the neighborhood. With LoG images, the sets of zero-crossing pixels tend to be the image-edges. Therefore, computer programs detect edges using the zero-crossing of the LoG of an image.

- **Line Detection (Hough Transform):** A line can be described using polar coordinates:

  $$\rho = x*\cos(\theta) + y*\sin(\theta)$$

  Set of lines passing through each point Pi is represented as set of sine curves in (ρ, θ) space → multiple hits in space indicates presence of lines in original image. Use the Hough Transform to detect the lines of an edge detected image.

ALGORITHM DESCRIPTION: (*Discuss the specific algorithms or techniques used to implement the project. Pseudo code or flowcharts should be used where relevant.*)

I. **MATHEMATICAL MORPHOLOGY**

First, we load coins.bmp into the program's memory. Next, use thresholding to binarize the coins_grey.bmp image. Call the thresheld, binary image coins_thresh.bmp. Next, using a 3x3 square structural element, we dilate coins_thresh.bmp. Next, using an r=29 circular structural element, we erode the dilated coins_thresh.bmp image.

- **Dilation with square structural element.**
```
For Each Pixel
        If any pixel within SQUARE is black
                Set the central pixel to black.
```

- **Erosion with circular structural element**
```
For Each Pixel
        If any pixel within CIRCLE is white
                Set the central pixel to white.
```

- **Dilation with circular structural element**
```
For Each Pixel
        If any pixel within CIRCLE is black
                Set the central pixel to black.
```

- **Erosion with square structural element**
```
For Each Pixel
        If any pixel within SQUARE is white
                Set the central pixel to white.
```

II. **CONNECTED COMPONENT LABELING**

To count the number of coins in coins.bmp, I used the connected component labeling algorithm below. This algorithm computes the connected components of a binary image.

- **Connected Component Labeling**

  B is the original binary image (coins.bmp)

  LB will be the labeled connected component image.

```
void recursive_connected_components (B, LB) {
        LB = negate(B);
        label = 0;
        find_components(LB, label);
        print(LB);
}
```

```
void find_components(LB, label) {
        for (L=0; L < MaxRow; L++) {
                for (P=0; P<MaxCol; P++) {
                        if (LB[L,P] == -1) {
                                label++;
                                search(LB, label, L, P);
                        }
                }
        }
}
void search (LB, label, L, P) {
        LB[L,P] = label;
        Nset = neighbors(L,P);
        for each LB[L',P'] in Nset {
                if (LB[L',P'] == -1) {
                        search(LB, label, L', P');
                }
        }
}
```

III. LINE DETECTION

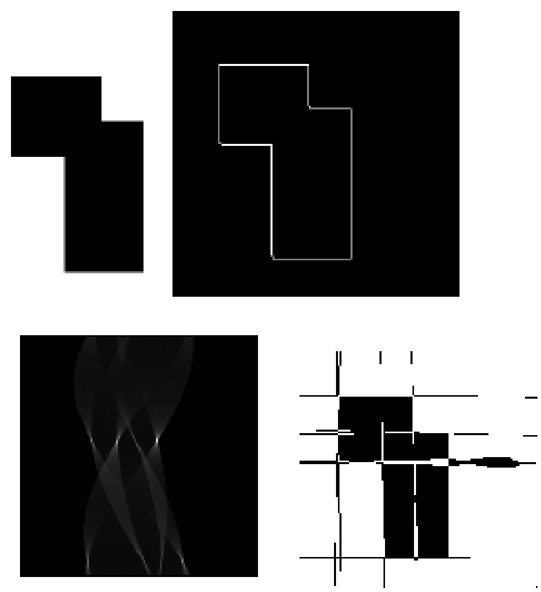
On B2bomber.bmp, to detect lines, apply the Hough Transform to the image.

First, on B2bomber.bmp, apply the Laplacian of Gaussian (LoG) to compute an edged image. B2bomber_edged.bmp.

For each edge detected edge point, increment all the possible values of angle θ and compute the corresponding values of ρ using the line equation: $\rho = x*\cos(\theta) + y*\sin(\theta)$

Accumulate these values in the (ρ, θ) space. At the end, identify the cells that contain the most counts as cells corresponding to the lines.

The images to the right represent the line detection algorithm pictorially. The original image (top left). The edge detected image (top right). The (ρ, θ) space (bottom left). The original image + the detected lines of the (ρ, θ) space (bottom left).



IV. CORNER DETECTION

Corners in gray-scale images occur in region of rapidly changing intensity levels → it is useful to have operators that detect corners directly without locating edges.

In this project, I detect the corners of the B2bomber.bmp image using the Zuniga-Haralick corner detector: based on approximation of image function by cubic polynomial.

On grey scale B2bomber.bmp, we compute the Z-H detector response for each pixel (i,j) such that :

$$ZH(i,j) = \frac{-2(c_2^2 c_6 - c_2 c_3 c_5 - c_3^2 c_4)}{(c_2^2 + c_3^2)}$$

The coefficients c_2, c_3, c_4, c_5, c_6 at each pixel location (I,j) may be computed by applying convolution masks.

Mask for c_2

31	-44	0	44	-31
-5	-62	0	62	5
-17	-68	0	68	17
-5	-62	0	62	5
31	-44	0	44	-31

Mask for c_3

-31	5	17	5	-31
44	62	68	62	44
0	0	0	0	0
-44	-62	-68	-62	-44
31	-5	-17	-5	31

Mask for c_4

2	-1	-2	-1	2
2	-1	-2	-1	2
2	-1	-2	-1	2
2	-1	-2	-1	2
2	-1	-2	-1	2

Mask for c_5

4	2	0	-2	-4
2	1	0	-1	-2
0	0	0	0	0
-2	-1	0	1	2
-4	-2	0	2	4

Mask for c_6

2	2	2	2	2
-1	-1	-1	-1	-1
-2	-2	-2	-2	-2
-1	-1	-1	-1	-1
2	2	2	2	2

The response ZH(i,j) is then computes and compared to a threshold. Exceeding the threshold is the detected corner points.

<u>RESULTS & ANALYSIS:</u> (*Provide the results of the work. Analyze the results. Determine if the results are as expected. Discuss. The results should include outputs of the program (e.g., processed images, tables, useful data).*)

I. MATHEMATICAL MORPHOLOGY

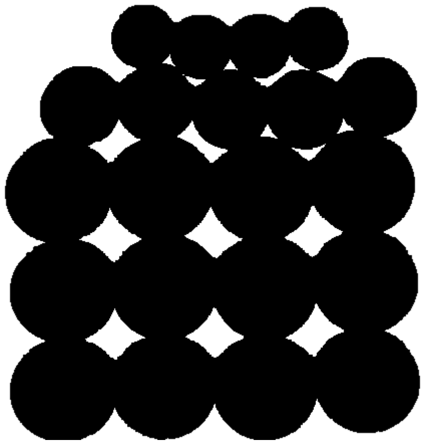
1. Perform an appropriate thresholding on the coins.bmp so that the image contains only black (foreground) and white (background).

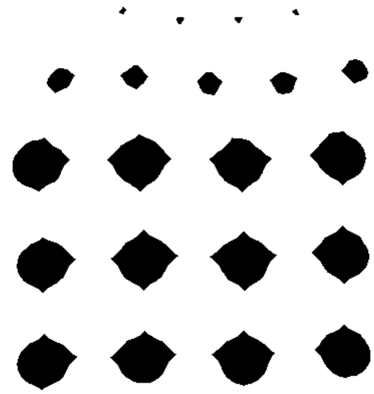
Coins_grey.bmp

Coins_thresh.bmp

2. Then, select an appropriate structure element and appropriate mathematical morphological operators (e.g., closing, opening) to operate on the thresholded coins.bmp such that the coins are well separated from one another. Show the results at each step.

    1. Dilate coins_thresh.bmp (using a 3x3 square structural element) to get coins_dilate.bmp.
    2. Erode coins_dilate.bmp (using a r=29 circular element) to get coins_erode.bmp.


Coins_dilate.bmp

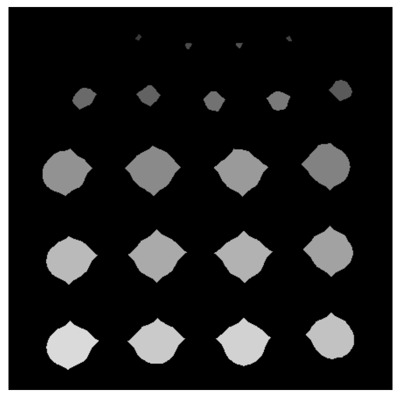

coins_erode.bmp

(Note: It took trial and error to find the correct structural elements)

II. CONNECTED COMPONENT LABELING

1. Apply the recursive connected component labeling on the result of the above mathematical morphology to label the coins. From this, determine the number of coins. Show the image result with different colors or shades of the coins (each coin has a distinct color or shade).

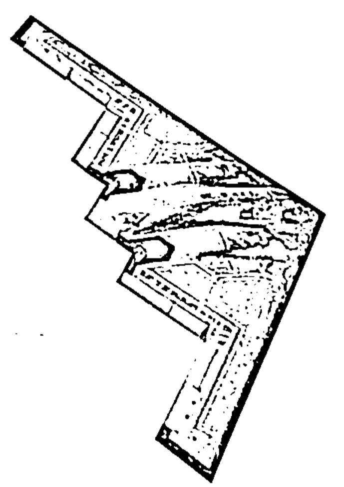
    1. Apply the recursive connected component labeling algorithm on coins_erode.bmp.
    2. Number of coins = 21
    3. Color each set of connected components a different shade of grey: coins_color_coded.bmp


Coins_color_coded.bmp

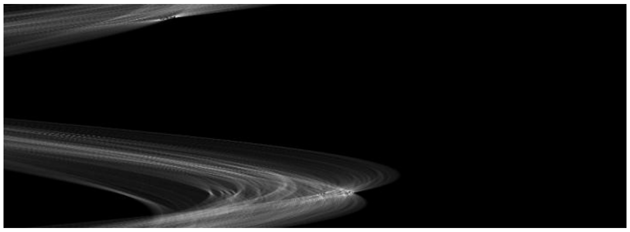
III. LINE DETECTION

1. Perform an edge detection on the gray level B2Bomber image. B2bomber.bmp → B2bomber_LoG_Edge.bmp.

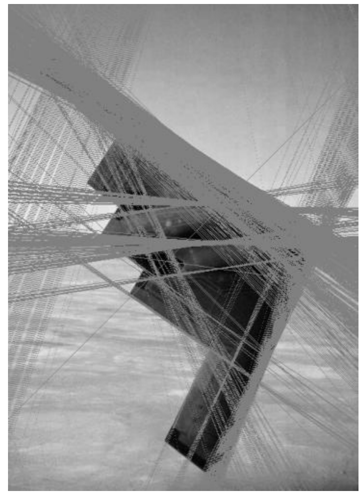

B2bomber.bmp


B2bomber_LoG_Edge.bmp

2. Then, apply the Hough Transform to detect the lines present in the edge detected image.
    1. Compute the (ρ, θ) space for B2bomber_LoG_Edge.bmp. Plot the (ρ, θ) space as B2bomber_lines.bmp.
    2. Draw significant points from the (ρ, θ) space as lines on-top the original image (B2bomber_lines_final.bmp).



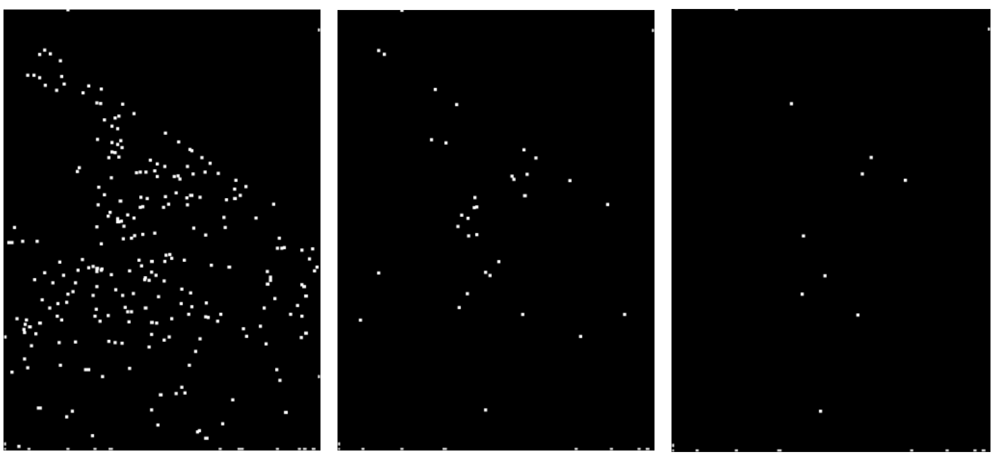
B2bomber_lines.bmp



B2bomber_lines_final.bmp

IV. CORNER DETECTION

1. Apply any corner detector to detect corner points on the gray level B2Bomber.
    1. Apply the Zuniga-Haralick corner detector to B2bomber_grey.bmp.
    2. Dilate the detected corner points.



<u>CONCLUSION:</u> (*Provide a brief conclusion of the project.*)

- Dilation and erosion are mathematical operations used to enhance images. Dilation and erosion require a structural element. Common structural elements include circle and square. Dilation followed by erosion is also known as closing. We closed coins.bmp to separate the coins from one another. Then, we performed the connected component labeling algorithm to count the number of coins in the image (21 coins).

- Next, we detected lines in an image. Using the Hough Transform, we transformed B2bomber.bmp into the (ρ, θ) space. Significant points in the (ρ, θ) space indicated lines of the B2bomber image. We plotted the detected lines onto the original image: the Hough Transform accurately detects edges.

- We used the Zuniga-Haralick corner detector to detect the corners of B2bomber.bmp.