```python
import cv2
import numpy as np
from tabulate import tabulate

# Boundary Detecting
def find_boundary(binary_image):
    kernel = np.ones((3,3), np.uint8)
    eroded = cv2.erode(binary_image, kernel, iterations=1)
    boundary = binary_image - eroded
    return boundary

# Perimeter, Area, Max_Diameter Counting
def calculate_perimeter(binary_image):
    kernel = np.ones((3,3), np.uint8)
    eroded = cv2.erode(binary_image, kernel, iterations=1)
    boundary = binary_image - eroded
    perimeter = np.count_nonzero(boundary)
    return perimeter
def calculate_area(binary_image):
    return np.sum(binary_image) / 255
```

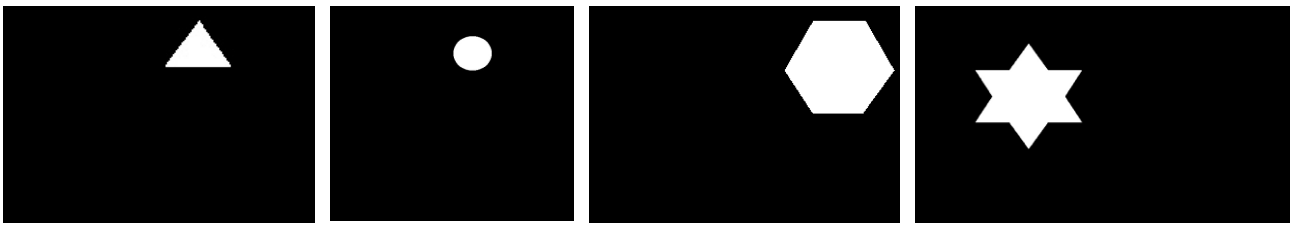
```python
def calculate_max_diameter(binary_image):
    contours, _ = cv2.findContours(binary_image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    max_diameter = 0
    for contour in contours:
        ellipse = cv2.fitEllipse(contour)
        major_axis = max(ellipse[1])
        minor_axis = min(ellipse[1])
        diameter_x = major_axis
        if diameter_x > max_diameter:
            max_diameter = diameter_x
    return max_diameter
```

```python
# Form Factor, Compactness, Descriptors, Compactness Calculating
def calculate_form_factor(area, perimeter):
    return (perimeter ** 2) / (4 * np.pi * area)
def calculate_roundness(area, perimeter):
    return (4 * np.pi * area) / (perimeter ** 2)
def calculate_compactness(area, perimeter, max_diameter):
    return (4 * np.pi * area) / (max_diameter ** 2)
```

```python
def calculate_descriptors(image):
    binary_image = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)[1]
    boundary = find_boundary(binary_image)
    perimeter = calculate_perimeter(boundary)
    area = calculate_area(binary_image)
    max_diameter = calculate_max_diameter(boundary)
    form_factor = calculate_form_factor(area, perimeter)
    roundness = calculate_roundness(area, perimeter)
    compactness = calculate_compactness(area, perimeter, max_diameter)
    return form_factor, roundness, compactness

train_image_paths = ["t1.jpg", "c1.jpg", "p1.png", "p3.jpg"]
test_image_paths = ["t2.jpg", "p2.png", "st.jpg", "c2.jpg"]

train_descriptors = []
test_descriptors = []
for path in train_image_paths:
    train_img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
    train_descriptors.append(calculate_descriptors(train_img))
for path in test_image_paths:
    test_img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
    test_descriptors.append(calculate_descriptors(test_img))
distances_matrix = np.zeros((len(test_descriptors), len(train_descriptors)))
```

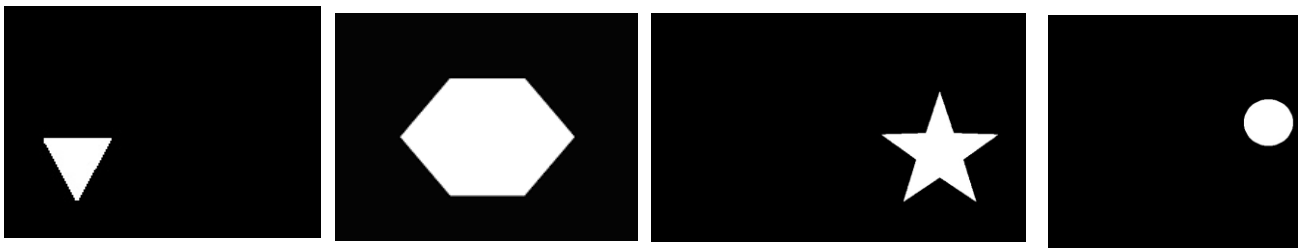
```python
for i, test_descriptor in enumerate(test_descriptors):
    for j, train_descriptor in enumerate(train_descriptors):
        distances_matrix[i][j] = np.sqrt(np.sum((np.array(test_descriptor) - np.array(train_descriptor)) ** 2))
row_headers = [f'Test {i + 1}' for i in range(len(test_image_paths))]
col_headers = [f'Train {i + 1}' for i in range(len(train_image_paths))]
table = tabulate(distances_matrix, headers=col_headers, showindex=row_headers, tablefmt='grid')
```

```python
best_matches = []
for i in range(len(test_image_paths)):
    best_match_index = np.argmin(distances_matrix[i])
    best_matches.append((i, best_match_index))

with open("results.txt", "w") as file:
    file.write("Distance Matrix:\n")
    file.write(table)
    file.write("\n\nTrain Images:\n1)Triangle\n2)Circle\n3)Hexagon\n4)Star")
    file.write("\n\nTest Images:\n1)Triangle\n2)Hexagon\n3)Star\n4)Circle")
    file.write("\n\nBest Matches:\n")
    for test_index, train_index in best_matches:
        file.write(f'Test {test_index + 1} best matches Train {train_index + 1}\n')
```



1) t1.jpg 2) c1.jpg 3) p1.jpg 4) p3.png

Figure 5.1: Train Images



1) t2.jpg 2) p2.png 3) st.jpg 4) c2.jpg

Figure 5.2: Test Images

Distance Matrix:

	Train 1	Train 2	Train 3	Train 4
Test 1	2.84995	11.6722	12.8857	13.536
Test 2	16.5509	2.06062	0.82934	1.47056
Test 3	19.2185	5.89976	4.92897	3.38371
Test 4	14.5544	0.0375575	1.19554	2.49

```
Best Matches:
Test 1 best matches Train 1
Test 2 best matches Train 3
Test 3 best matches Train 4
Test 4 best matches Train 2
```