МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования

«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра «Информатика и управление в
технических системах»

РАСЧЁТНО-ГРАФИЧЕСКАЯ РАБОТА

по дисциплине
«Основы технического зрения»

Вариант: 6

Выполнил:
студент группы МР/б-20-1-о
Горобей Р.А.

Проверил:
Жиляков П.В.

Севастополь
2024 г.

ПОСТАНОВКА ЗАДАЧИ И ИСХОДНЫЕ ДАННЫЕ

Цель работы:

Используя среду MATLAB и функции пакета Image Processing Toolbox:

- применить ранее изученный материал из практических работ;
- закрепить ранее изученные знания;
- получить умения и навыки по применению изученных в данном курсе знаний.

Вариант: 6

Порядок выполнения работы:

1. Загрузить исходное изображение Pic_pr3_1.bmp с заданными объектами.

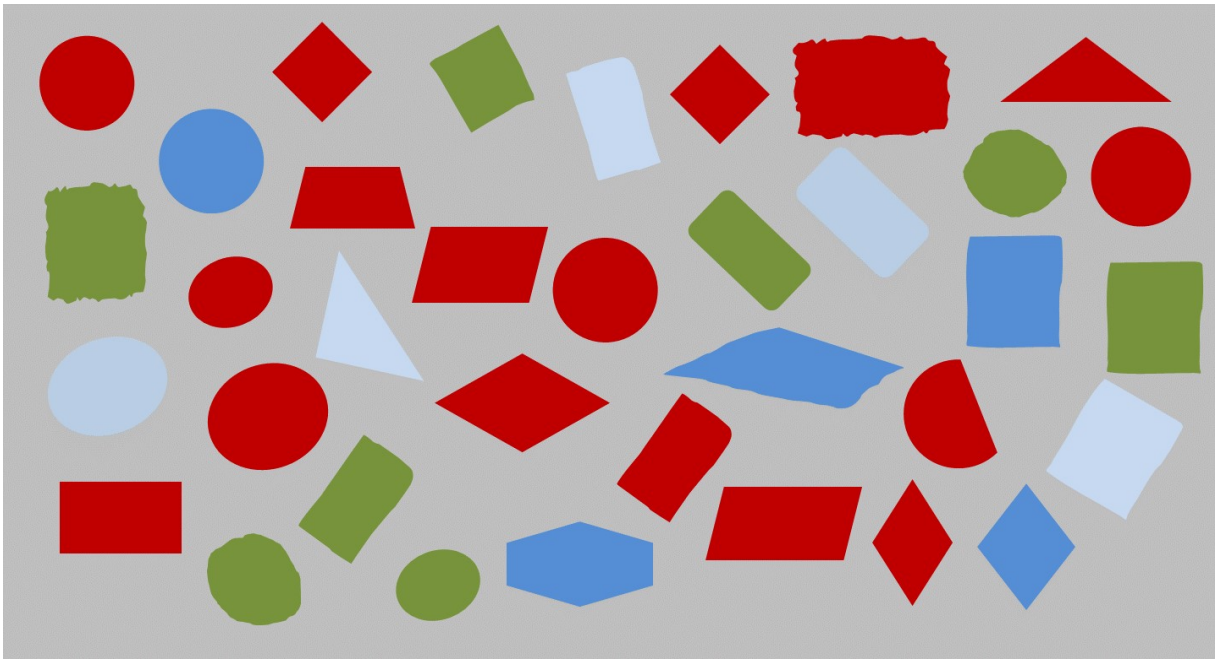

Рисунок 1 – Исходное изображение

2. Произвести сегментацию изображения. Определить и выделить:
   2.1. Наименьший по площади голубой объект на изображении;
   2.2. Наименьший по площади красный объект на изображении;
   2.3. Два зелёных объекта находящихся на минимальном расстоянии друг от друга;
   2.4. Зелёный объект с наибольшим углом наклона.

Ход работы:

Задача 1:

```
clear
% Загружаем изображение
originalImage = imread('Pic_pr3_1.bmp');

% Переводим в HSV для упрощённой сегментации по цвету
hsvImage = rgb2hsv(originalImage);

%Определяем пороги для голубого цвета
hueThresholdLow = 0.5;
hueThresholdHigh = 0.67;
saturationThresholdLow = 0.5;
valueThresholdLow = 0.5;

% Создаём маски для голубых обьектов
blueMask = (hsvImage(:,:,1) >= hueThresholdLow) & ...
          (hsvImage(:,:,1) <= hueThresholdHigh) & ...
          (hsvImage(:,:,2) >= saturationThresholdLow) & ...
          (hsvImage(:,:,3) >= valueThresholdLow);

% Производим морфологическое закрытие для избавления от щелей
se = strel('disk', 3);
blueMaskClosed = imclose(blueMask, se);

% Маркируем обьекты
[labels, numObjects] = bwlabel(blueMaskClosed, 8);

% Измеряем площадь и индекс наименьшего обьекта
objectMeasurements = regionprops(labels, 'Area', 'PixelIdxList');

% Выбираем наименьший обьект
[minArea, minAreaIndex] = min([objectMeasurements.Area]);

% Выделяем наименьший объект черной рамкой
smallestObjectBoundary = bwboundaries(labels == minAreaIndex);
boundary = smallestObjectBoundary{1};
imshow(originalImage);
hold on;
plot(boundary(:,2), boundary(:,1), 'k', 'LineWidth', 3); % Рисуем границы черным цветом

% Отображение площади на каждом голубом объекте
for k = 1:numObjects
    objectArea = objectMeasurements(k).Area;
    centroid = regionprops(labels == k, 'Centroid');
    text(centroid.Centroid(1), centroid.Centroid(2), num2str(objectArea), ...
         'Color', 'w', 'FontSize', 12, 'FontWeight', 'bold');
end
hold off;
```

Результат выполнения программы:

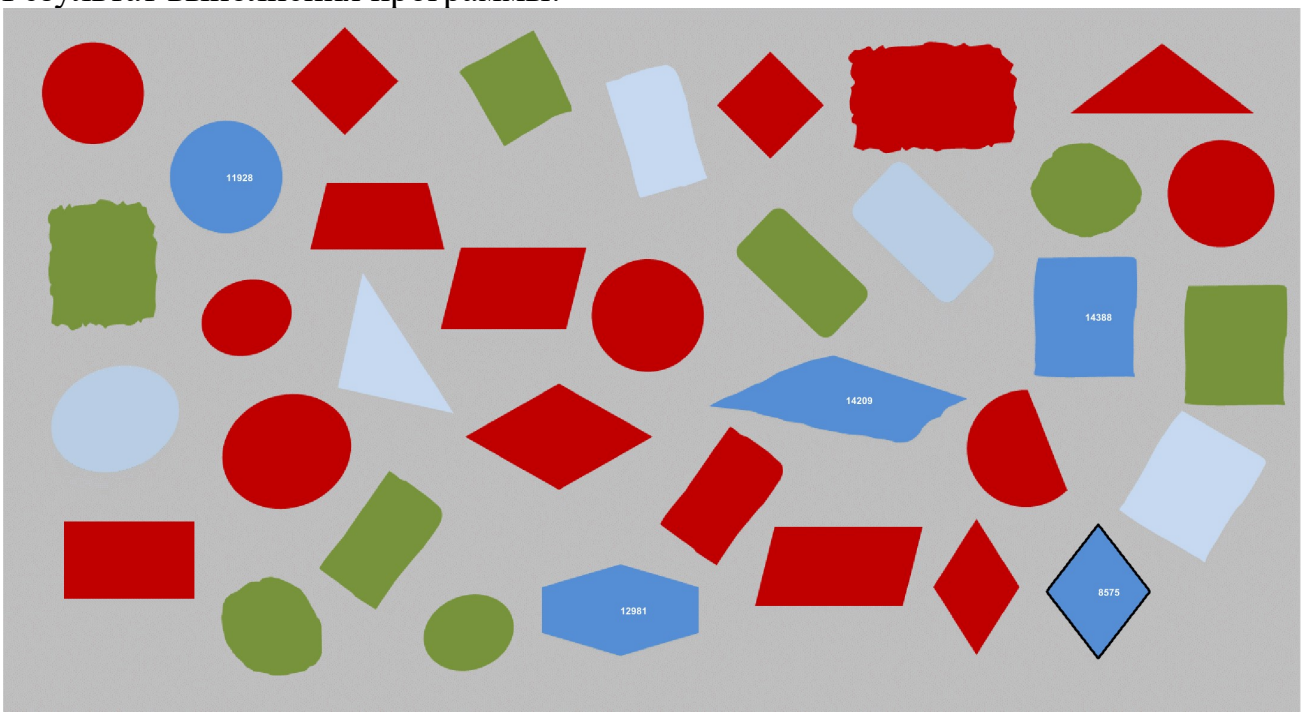

Рисунок 2 – результат выполнения 1 программы

Задача 2:

```
clear
% Загружаем изображение
originalImage = imread('Pic_pr3_1.bmp');

% Переводим в HSV для упрощённой сегментации по цвету
hsvImage = rgb2hsv(originalImage);

%Определяем пороги для красного цвета
hueThresholdLow = 0.0;
hueThresholdHigh = 0.1;
saturationThresholdLow = 0.5;
valueThresholdLow = 0.5;

% Создаём маски для красных обьектов
redMask = (hsvImage(:,:,1) >= hueThresholdLow) & ...
          (hsvImage(:,:,1) <= hueThresholdHigh) & ...
          (hsvImage(:,:,2) >= saturationThresholdLow) & ...
          (hsvImage(:,:,3) >= valueThresholdLow);

% Производим морфологическое закрытие для избавления от щелей
se = strel('disk', 3);
redMaskClosed = imclose(redMask, se);

% Маркируем обьекты
[labels, numObjects] = bwlabel(redMaskClosed, 8);

% Измеряем площадь и индекс наименьшего обьекта
```

```
objectMeasurements = regionprops(labels, 'Area', 'PixelIdxList');

% Выбираем наименьший обьект
[minArea, minAreaIndex] = min([objectMeasurements.Area]);

% Выделяем наименьший объект черной рамкой
smallestObjectBoundary = bwboundaries(labels == minAreaIndex);
boundary = smallestObjectBoundary{1};
imshow(originalImage);
hold on;
plot(boundary(:,2), boundary(:,1), 'k', 'LineWidth', 3); % Рисуем границы черным цветом

% Отображение площади на каждом красном объекте
for k = 1:numObjects
    objectArea = objectMeasurements(k).Area;
    centroid = regionprops(labels == k, 'Centroid');
    text(centroid.Centroid(1), centroid.Centroid(2), num2str(objectArea), ...
        'Color', 'w', 'FontSize', 12, 'FontWeight', 'bold');
end
hold off;
```

Результат выполнения программы:

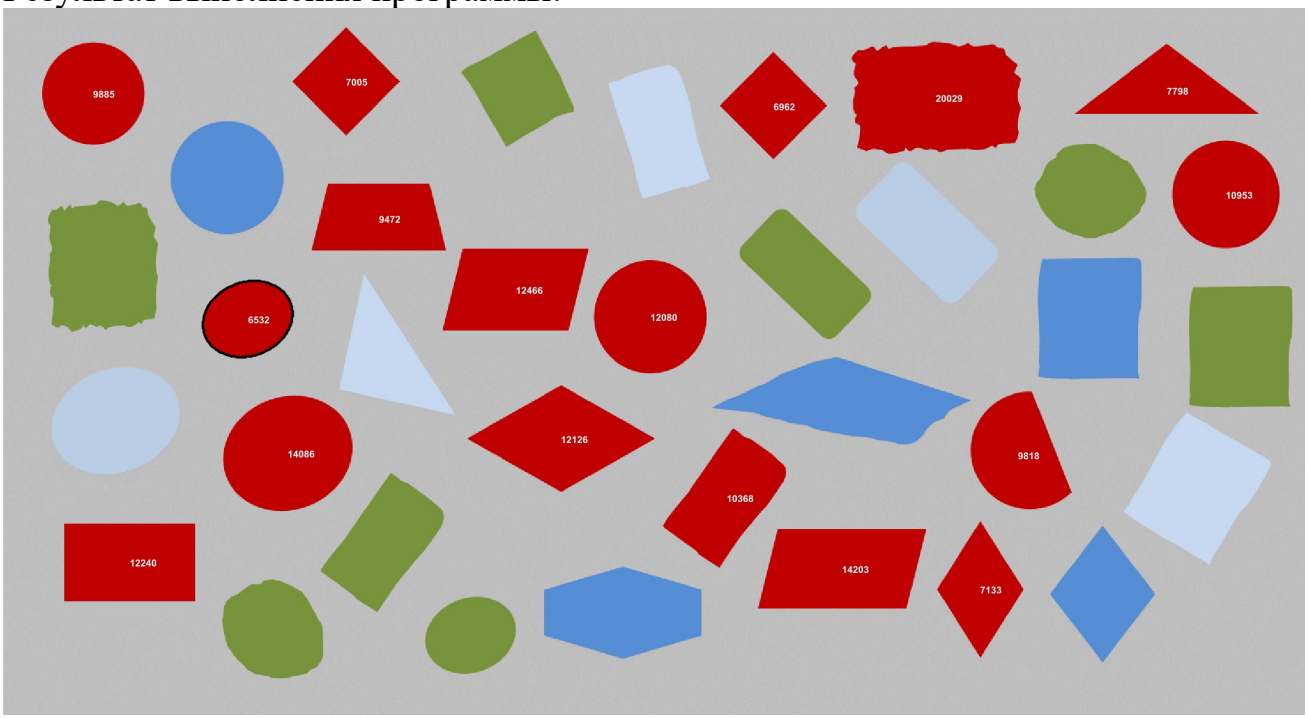

Рисунок 3 – результат выполнения 2 программы

Задача 3:

```
% Загрузка изображения 'Pic_pr3_1.bmp'
originalImage = imread('Pic_pr3_1.bmp');

% Переводим в HSV для упрощённой сегментации по цвету
hsvImage = rgb2hsv(originalImage);

%Определяем пороги для зелёного цвета
```

```
hueThresholdLow = 0.2;
hueThresholdHigh = 0.5;
valueThresholdLow = 0.1;

% Создаём маски для зелёных обьектов
greenMask = (hsvImage(:,:,1) >= hueThresholdLow) & ...
            (hsvImage(:,:,1) <= hueThresholdHigh) & ...
            (hsvImage(:,:,3) >= valueThresholdLow);

% Удаление шума с помощью морфологической операции
se = strel('disk', 3);
greenMaskCleaned = imopen(greenMask, se);

% Нахождение контуров зеленых объектов
[B,L] = bwboundaries(greenMaskCleaned, 'noholes');

imshow(originalImage);
hold on;

% Вычисление минимального расстояния между краями зеленых объектов
minDistance = inf;
closestObjects = [];

% Перебор всех пар зеленых объектов
for i = 1:length(B)
    for j = i+1:length(B)
        boundary1 = B{i};
        boundary2 = B{j};

        % Перебор всех точек первого объекта
        for p1 = 1:size(boundary1, 1)
            % Перебор всех точек второго объекта
            for p2 = 1:size(boundary2, 1)
                % Расчет расстояния между точками
                distance = norm(boundary1(p1,:) - boundary2(p2,:));
                % Обновление минимального расстояния и сохранение объектов
                if distance < minDistance
                    minDistance = distance;
                    closestObjects = [i, j];
                end
            end
        end
    end
end

% Вывод результатов
fprintf('Расстояние между двумя ближайшими зелёными обьектами: %f\n', minDistance);

% Проведение линии между ближайшими объектами
boundary1 = B{closestObjects(1)};
boundary2 = B{closestObjects(2)};

% Нахождение крайних точек ближайших объектов
[minDist, minIndex1] = min(pdist2(boundary1, boundary2, 'euclidean'), [], 1);
[~, minIndex2] = min(minDist);

% Координаты ближайших точек на каждом объекте
point1 = boundary1(minIndex1(minIndex2), :);
point2 = boundary2(minIndex2, :);
```

```
% Рисование линии между ближайшими точками двух объектов
line([point1(2), point2(2)], [point1(1), point2(1)], 'Color', 'black', 'LineWidth', 3);
```

Результат выполнения программы:

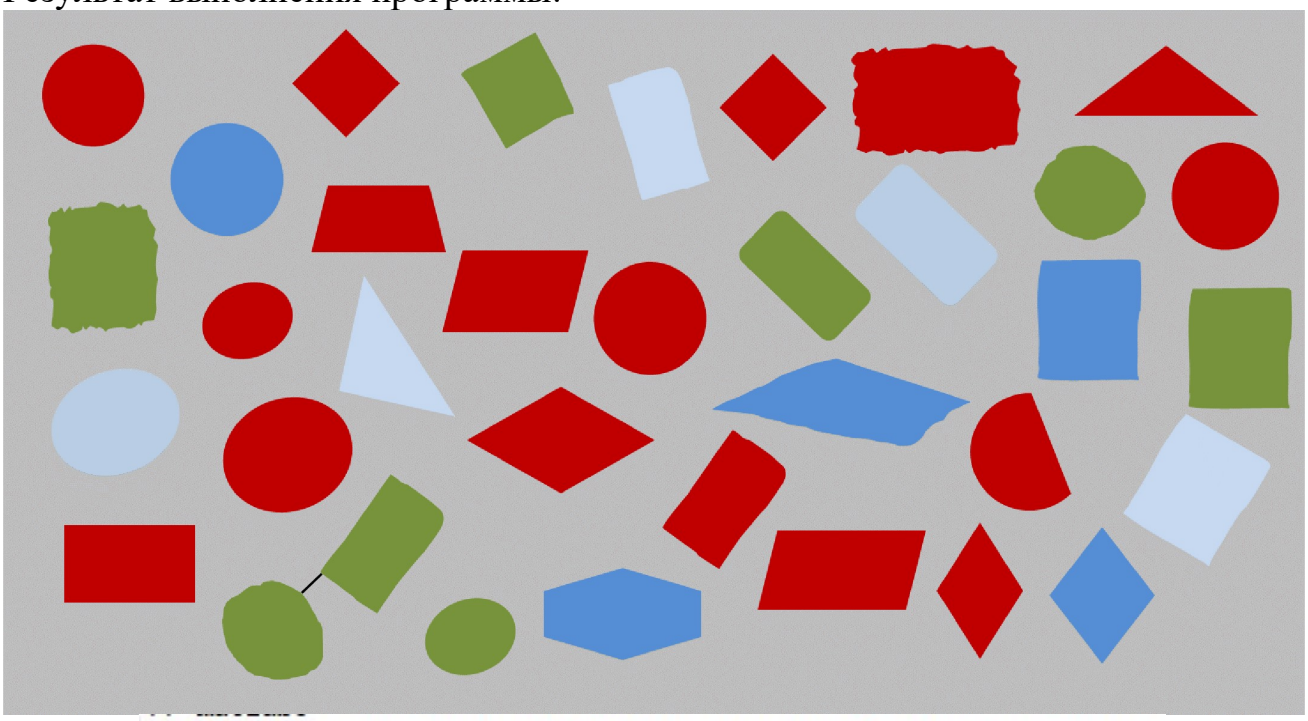

Рисунок 4 – результат выполнения 3 программы

Задача 4:

```
% Загрузка изображения 'Pic_pr3_1.bmp'
originalImage = imread('Pic_pr3_1.bmp');

% Преобразование изображения в пространство цветов HSV
hsvImage = rgb2hsv(originalImage);

%Определяем пороги для зелёного цвета
hueThresholdLow = 0.2;
hueThresholdHigh = 0.5;
valueThresholdLow = 0.1;

% Создаём маски для зелёных обьектов
greenMask = (hsvImage(:,:,1) >= hueThresholdLow) & ...
            (hsvImage(:,:,1) <= hueThresholdHigh) & ...
            (hsvImage(:,:,3) >= valueThresholdLow);

% Удаление шума с помощью морфологической операции
se = strel('disk', 3);
greenMaskCleaned = imopen(greenMask, se);
```

```
% Отображение изображения
imshow(originalImage);
hold on;
title('Зелёные обьекты');

% Проверяем параметр Orientation у каждого обьекта и пишем его
properties = regionprops(L, 'Orientation', 'Centroid');

for k = 1:length(properties)
    orientation = properties(k).Orientation;
    centroid = properties(k).Centroid;
    text(centroid(1), centroid(2), sprintf('Угол наклона: %.2f', orientation), ...
        'Color', 'black', 'FontSize', 12, 'FontWeight', 'bold');
end
% Нахождение объекта с наибольшим углом наклона
[maxOrientationValue, maxOrientationIndex] = max([properties.Orientation]);

% Выделение объекта с наименьшим углом наклона чёрной рамкой
maxBoundary = B{maxOrientationIndex};
plot(maxBoundary(:,2), maxBoundary(:,1), 'k', 'LineWidth', 3);

hold off
```

Результат выполнения программы:

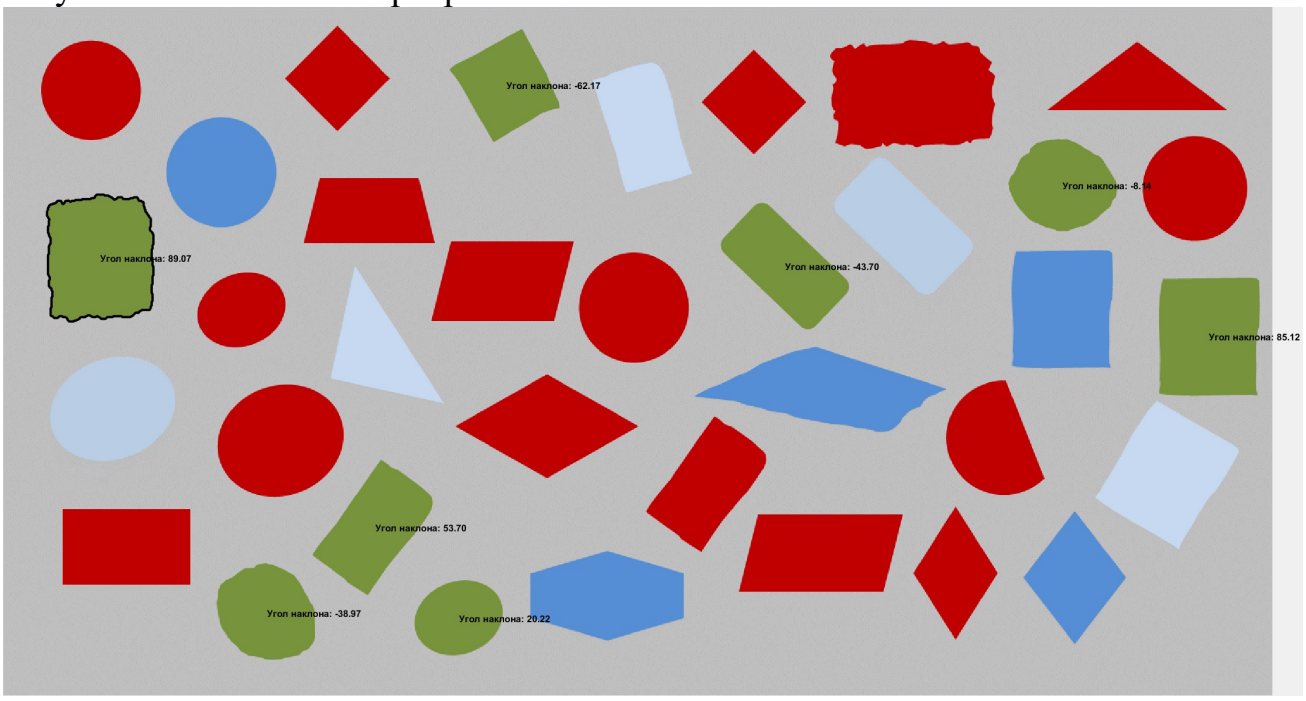

Рисунок 5 – результат выполнения 4 программы

Вывод: В процессе работы с пакетом **Image Processing Toolbox** в **Matlab** были изучены, применены и освоены различные методы обработки изображений. Это включало в себя такие операции, как:

1. Улучшение качества изображения
2. Сегментация
3. Морфологические преобразования.
4. Обнаружение объектов.