



НИУ ИТМО

Отчет по лабораторной работе №6

«Морфологический анализ изображений»

Выполнили:

Александр Иванов, Ф ТЕХ.ЗРЕНИЕ 1.1

Ани Аракелян, ТЕХ.ЗРЕНИЕ 1.1

Никита Братушка, ТЕХ.ЗРЕНИЕ 1.3

Преподаватель:

Шаветов С. В.

Санкт-Петербург, 2024

# **Содержание**

<b>1. Базовые морфологические операции</b>	<b>3</b>
1.1. Исходное изображение . . . . .	3
1.2. Результаты . . . . .	5
<b>2. Разделение объектов</b>	<b>6</b>
2.1. Исходное изображение . . . . .	6
2.2. Программа на языке MATLAB . . . . .	6
2.3. Результаты . . . . .	7
<b>3. Сегментация</b>	<b>8</b>
3.1. Исходное изображение . . . . .	8
3.2. Программа на языке Python . . . . .	8
3.3. Результат . . . . .	9
<b>4. Выводы</b>	<b>13</b>
<b>5. Ответы на вопросы</b>	<b>13</b>
<b>6. P.S.</b>	<b>14</b>

# 1. Базовые морфологические операции

## 1.1. Исходное изображение

Для первого задания выберем фотографию Яна Берри, сделанную в разгар событий Пражской весны:

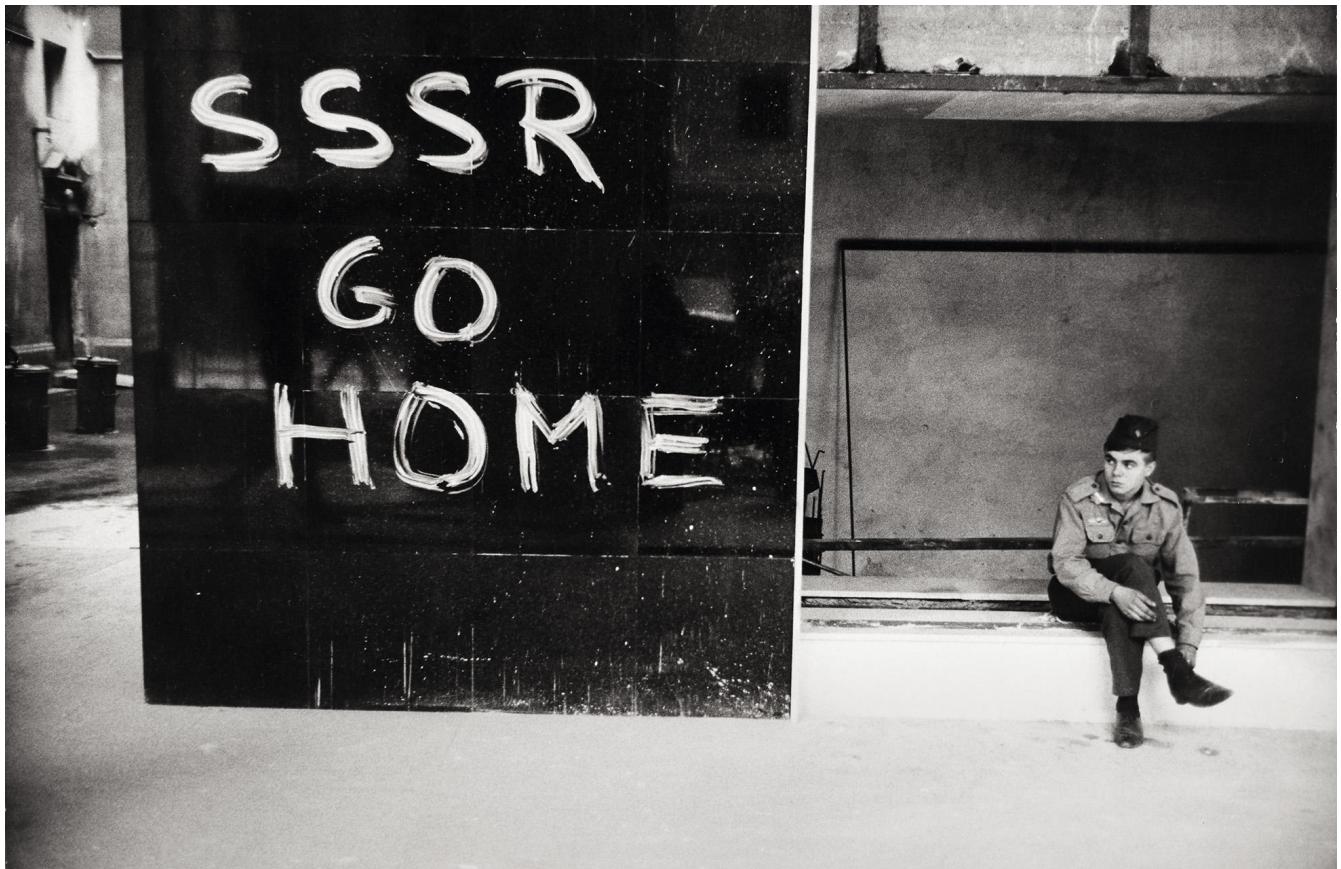


Рис. 1: Ян Берри. Молодой советский солдат отдыхает рядом с плакатом «СССР, возвращайся домой» на Вацлавской площади. Прага, Чехословакия. 1968

Нашей задачей будет минимизация дефектов формы на надписи *SSSR GO HOME*. В этом нам поможет *Python*. Для начала загрузим изображение и выберем область изображения с надписью для выполнения преобразований.

```
# loading grayscale image
src_img = cv.imread('source_images/Ian_Berry_A young_Russian
soldier.Jpeg', 0)
assert src_img is not None, "File could not be read"
result = np.copy(src_img)
# selecting specific area
spec_area = src_img[0:735, 200:1080]
```

Листинг 1: Исходный код для считывания изображения



Рис. 2: Надпись, которая подвергнется преобразованиям

С помощью эрозии избавимся от лишних точек на стене и выступов на буквах. Далее применим дилатацию от избавления от «внутренних дырок» и затем эрозию для избавления от частиц, появившихся в результате дилатации, и возвращения буквам их исходной толщины.

```
# morphological operations
disk = cv.getStructuringElement(cv.MORPH_ELLIPSE, (3, 3))
erosion = cv.erode(spec_area, disk, iterations=1)
dilation = cv.dilate(erosion, disk, iterations=7)
erosion2 = cv.erode(dilation, disk, iterations=5)
```

Листинг 2: Исходный код для преобразования надписи

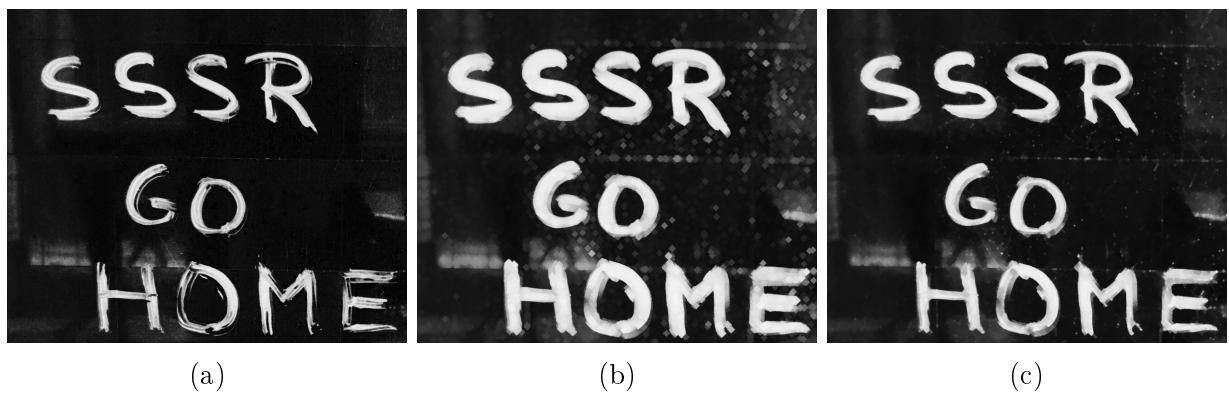


Рис. 3: Избавление от дефектов: (а) результат применения эрозии, (б) результат применения дилатации 7 раз, (с) результат применения эрозии 5 раз

## 1.2. Результаты

После преобразований необходимо вернуть «новую» надпись в исходное изображение.

```
# displaying transformed image  
result[0:735, 200:1080] = erosion2  
display_image('Result', result, 1)
```

Листинг 3: Исходный код для получения итогового изображения



Рис. 4: Результат применения базовых морфологических операций

## 2. Разделение объектов

### 2.1. Исходное изображение

Обратимся к кругам и их производным. Исходное изображение:

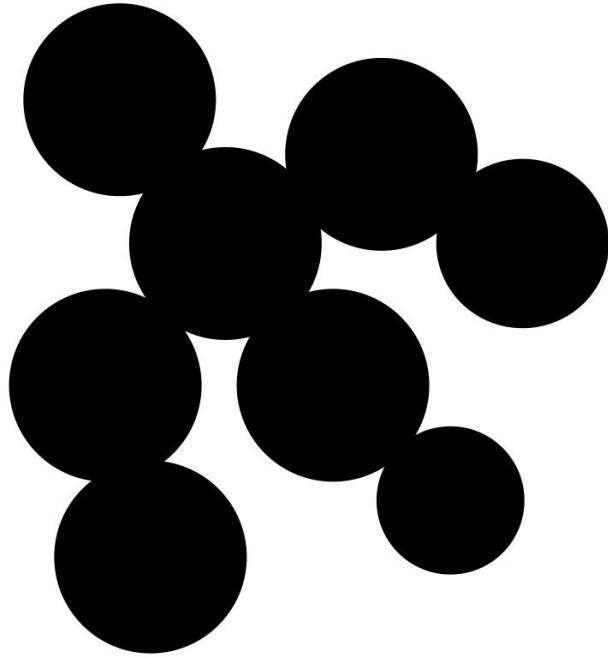


Рис. 5: Круги

Для выполнения этого задания воспользуемся *MATLAB*.

### 2.2. Программа на языке MATLAB

```
src_img = imread("circles.jpg");
gray_img = im2gray(src_img);
BW = imbinarize(gray_img);
BW = ~BW;
imwrite(BW,"binary_inv.jpg");
BW2 = bwmorph(BW,'erode',45);
imwrite(BW2,"erosed.jpg");
BW2 = bwmorph(BW2,'thicken',Inf);
imwrite(BW2,"boundaries.jpg");
BW = ~(BW & BW2);
imwrite(BW,"result.jpg");
```

Листинг 4: Исходный код программы для разделения объектов

## 2.3. Результаты

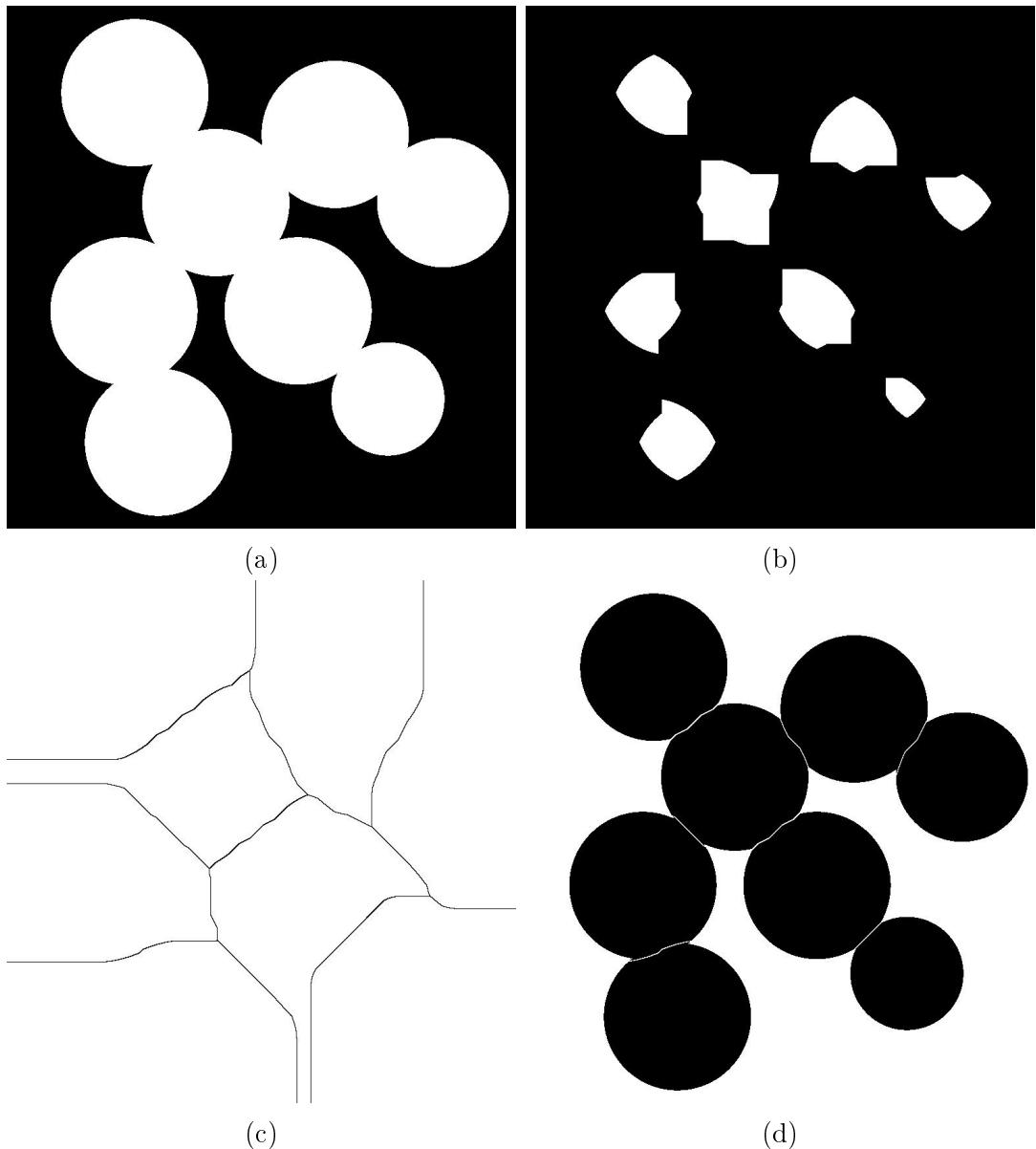


Рис. 6: Разделение «склеенных объектов»: (а) инвертированное бинарное изображение, (б) эрозия бинарного изображения, (с) расширение объектов, (д) результат разделения

### 3. Сегментация

#### 3.1. Исходное изображение

Для последнего задания выберем следующее изображение:



Рис. 7: Шары для игры в бильярд

#### 3.2. Программа на языке Python

```
def bwareaopen(image, dim, conn=8):
    assert image.ndim == 2, None
    # Find all connected components
    num, labels, stats, centers =
    cv.connectedComponentsWithStats(image, connectivity=conn)
    # Check size of all connected components
    for i in range(num):
        if stats[i, cv.CC_STAT_AREA] < dim:
            image[labels == i] = 0
    return image
```

```

# loading an image
src_img = cv.imread('source_images/snooker.jpg')
# converting it to binary
gray_img = cv.cvtColor(src_img, cv.COLOR_BGR2GRAY)
ret, i_bw = cv.threshold(gray_img, 0, 255, cv.THRESH_BINARY +
cv.THRESH_OTSU)
# deleting tiny connected components
i_bw = bwareaopen(i_bw, 20, 8)
disk = cv.getStructuringElement(cv.MORPH_ELLIPSE, (5, 5))
i_bw = cv.morphologyEx(i_bw, cv.MORPH_CLOSE, disk)
# display_image('Binary_closed', i_bw, 1)

# finding foreground
i_fg = cv.distanceTransform(i_bw, cv.DIST_L2, 5)
ret1, i_fg = cv.threshold(i_fg, 0.6 * i_fg.max(), 255, 0)
i_fg = i_fg.astype(np.uint8)
ret2, markers = cv.connectedComponents(i_fg)
# display_image('Fg', i_fg, 1)

# Find background location
i_bg = np.zeros_like(i_bw)
markers_bg = markers.copy()
markers_bg = cv.watershed(src_img, markers_bg)
i_bg[markers_bg == -1] = 255
# display_image('Bg', i_bg, 1)

# Define undefined area
i_unk = cv.subtract(i_bg, i_fg)
# Define all markers
markers = markers + 1
markers[i_unk == 255] = 0
# Do watershed
# Prepare for visualization
markers = cv.watershed(src_img, markers)
# display_image('UND', i_unk, 1)
markers_jet = cv.applyColorMap((markers.astype(np.float32)*255/(ret2 +
1)).astype(np.uint8), cv.COLORMAP_JET)
display_image('Jet_fg_markers', markers_jet, 1)
src_img[markers == -1] = (0, 0, 255)
# display_image('result', src_img, 1)

```

Листинг 5: Исходный код программы для сегментации изображения

### 3.3. Результат

Для начала необходимо получить бинарное изображение, предварительно удалив лишние детали.

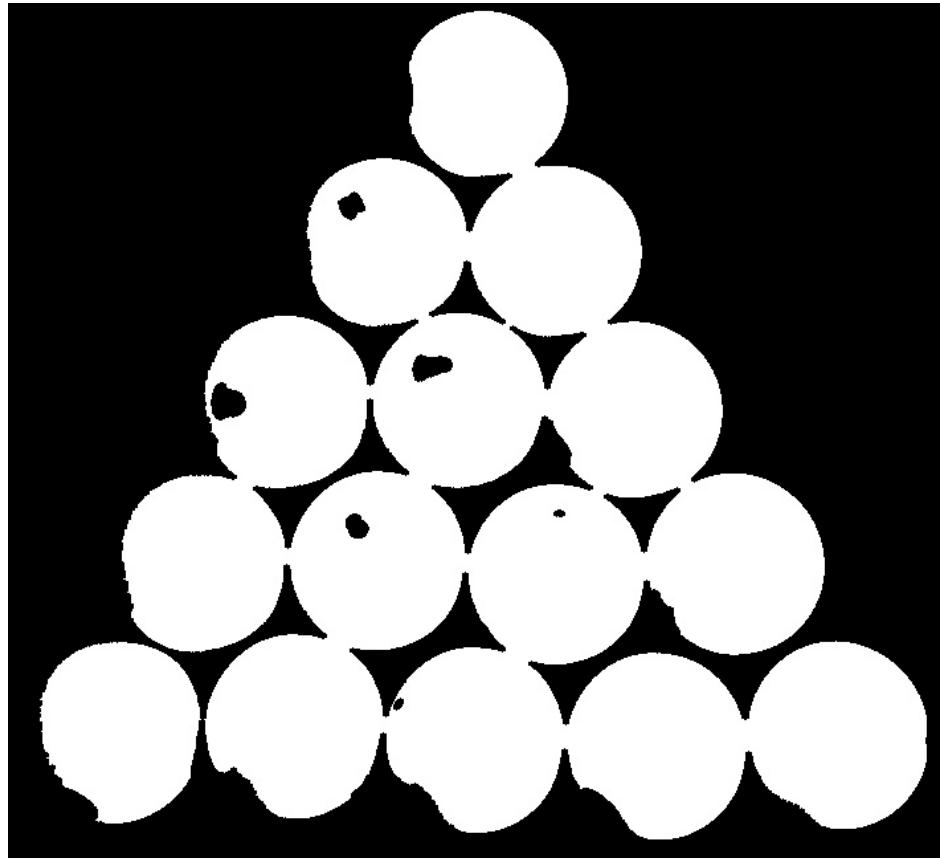


Рис. 8: Бинарное изображение

Далее воспользуемся преобразованием евклидова расстояния и после дополнительной фильтрации получим маркеры переднего плана:

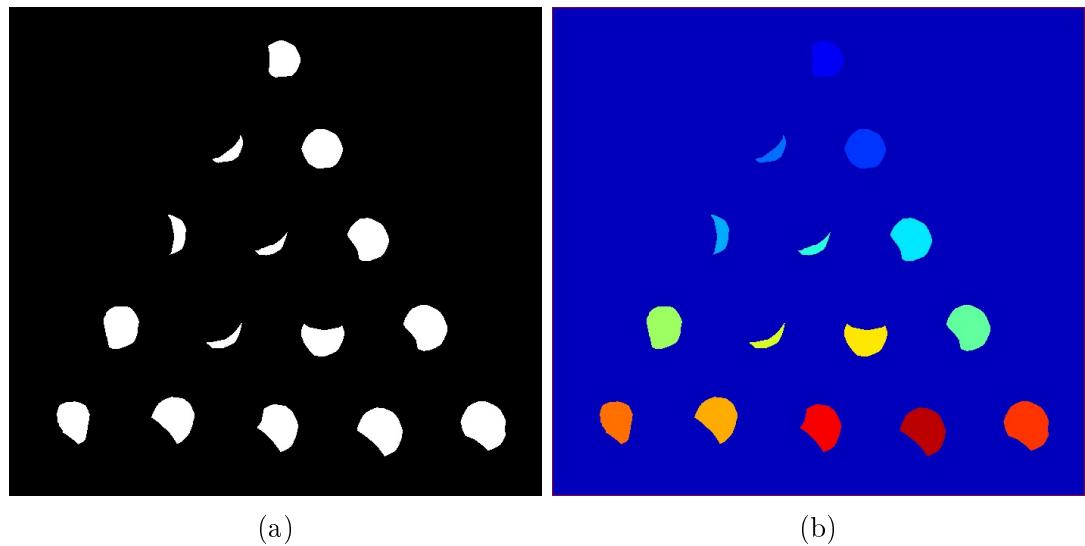


Рис. 9: Определение маркеров переднего плана: (а) область маркеров переднего плана, (б) маркеры переднего плана

Далее получим маркеры фона и неопределенной области:

Наконец, получим результаты сегментации:

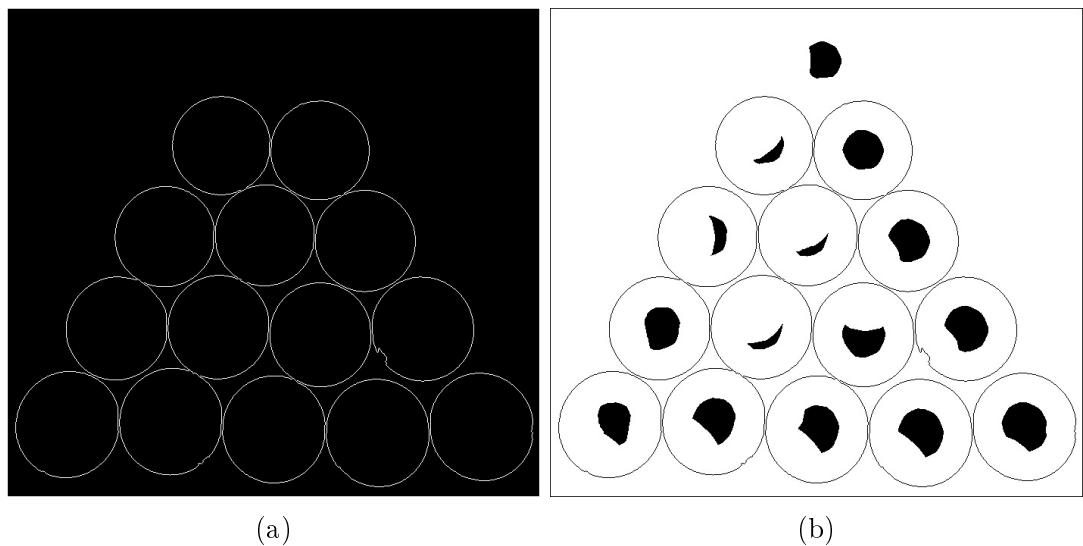


Рис. 10: Определение маркеров: (а) фона, (б) неопределенной области

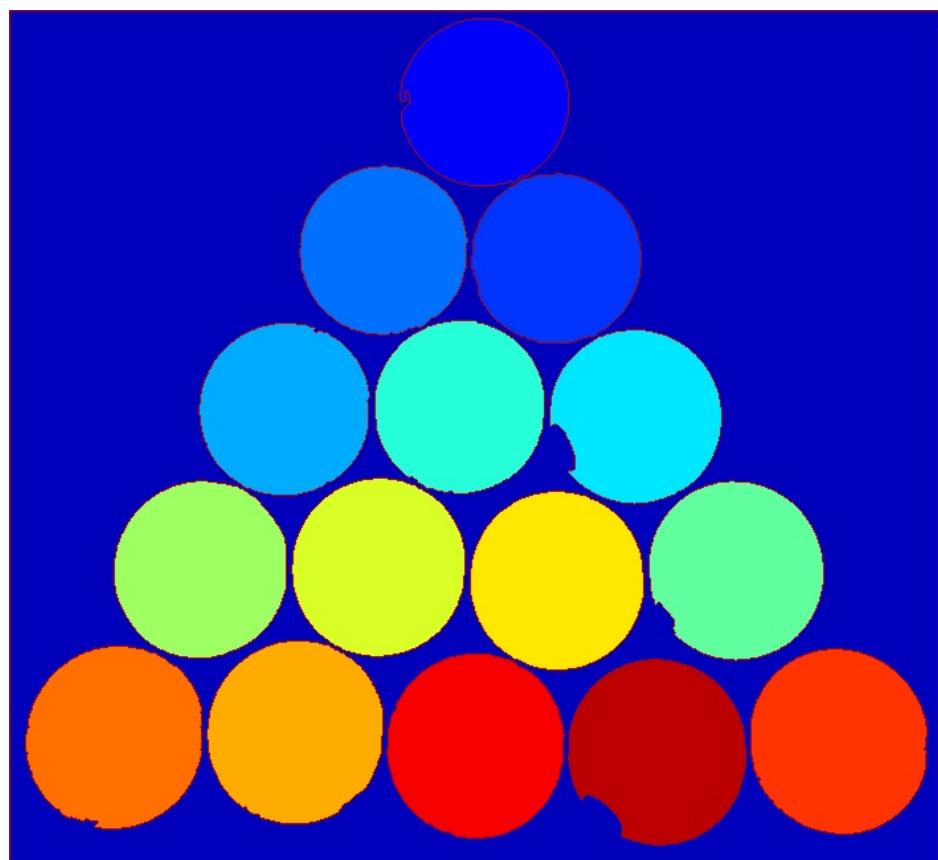


Рис. 11: Результат сегментации



Рис. 12: Результат сегментации, наложенный на исходное изображение

## 4. Выводы

В результате выполнения работы мы познакомились с морфологическими операциями, а также рассмотрели задачи, для решения которых они применяются. С помощью Морфологических операций нам удалось минимизировать дефекты формы (рис. 4), разделить «склеенные» объекты (рис. 6d) и сегментировать изображение (рис. 12).

Полный код программы и изображения можно найти на [GitHub](#).

## 5. Ответы на вопросы

Q1. Включает ли результат открытия в себя результат закрытия?

A1. Нет. Правильнее будет сказать, что результат **закрытия** включает в себя результат **открытия**

Q2. Какой морфологический фильтр необходимо применить, чтобы убрать у объекта выступы?

A2. Для того, чтобы убрать лишние выступы, можно использовать **эрозию**, однако это приведет к уменьшению размера объекта. Поэтому можно также воспользоваться операцией **открытия**, благодаря которой мы сможем избавиться от выступов и сохранить размеры объекта. Единственный недостаток этого подхода – острые углы будут становиться более округлыми.

Q3. Каким образом с помощью морфологических операций можно найти контур объекта?

A3. Для нахождения контура объекта нам необходимо, в первую очередь, получить бинарное изображение этого объекта. Далее мы должны вычесть (по правилам теории множеств) из результата **дилатации** исходное изображение, получив при этом внешний контур объекта. Для получения внутреннего контура необходимо из исходного изображения вычесть результат **эрозии**.

Q4. Что такое морфология?

A4. Дословно слово **морфология** переводится как наука о форме. Вообще говоря, морфология – это теория анализа и обработки геометрических фигур, основанная на теории множеств и топологии.

## 6. P.S.

Так это лабораторная знаменует собой окончание курса, то наша команда, в качестве благодарности, приготовила свои изображения:



