

Отчет по лабораторной работе №2

**Изучение и освоение методов обработки
и сегментации изображений**

Каратыщев Дмитрий Иванович

Май 2024

1 Постановка задачи

Необходимо разработать и реализовать программу для работы с изображениями фишек игрового набора «Тримино», обеспечивающую:

1. Ввод и отображение на экране изображений в формате BMP;
2. Сегментацию изображений на основе точечных и пространственных преобразований;
3. Поиск фишек на картинках
4. Классификацию фишек на картинках

2 Описание данных

В качестве входных данных предлагаются 9 цветных изображений различной сложности. Сложность определяется фоном, на котором расположены фишечки, количеством и расположением фишечек.

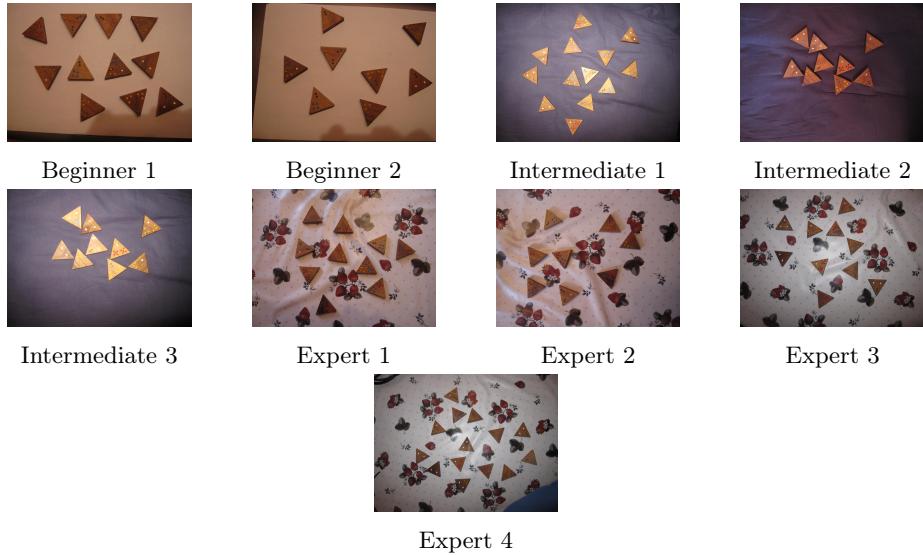


Рис. 1. Входные данные

Входные изображения подаются в формате BMP24. Выходом программы должен быть текстовый файл, в котором каждая запись описывает положение и код фишке в формате:

N – количество фишек на картинке
 $X, Y; m1, m2, m3;$

Здесь (X, Y) - координаты центра фишке на изображении. $m1, m2, m3$ - количество точек в углах треугольника.

3 Описание метода решения

Рассмотрим этапы обработки и сегментации входного изображения. Из предложенных уровней сложности были выполнены Beginner и Intermediate.

1. Определение типа изображения по среднему значению всех пикселей по каждому каналу.
2. Сегментация фишек тримино через бинаризацию и выделение контуров с их дальнейшим заполнением. Каждое изображение переводится в *hsv* формат, а далее выделяется канал *v*. После этого оператором Canny выделяются границы, и дальше происходит заливка полученных контуров.
3. Сегментация точек у каждого тримино по отдельности и подсчёт связных компонент. Было замечено, что цвет точки однозначно определяет количество точек в одном углу. Поэтому реализован подсчёт точек

каждого из 5 цветов. Пороги для бинаризации подбирались для формата bgr.

4. Поиск центроидов по контурам, соответствующим каждой фишке набора «Тримино»
5. Визуализация результата сегментации и подсчёта точек на каждой из фишек

4 Описание программной реализации

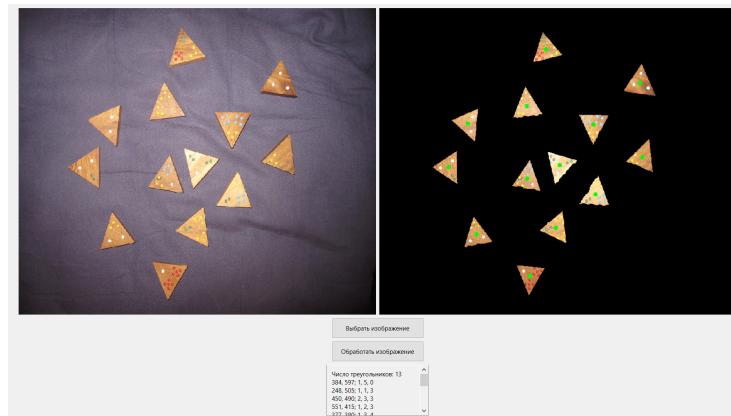


Рис. 2: Графический интерфейс

Для написания приложения использовался язык программирования Python с библиотеками PyQt6 для написания графического пользовательского интерфейса; пипму и опенсв для сегментации, бинаризации изображений, поиска связных компонент и нахождения центроидов. Весь программный код и релиз приложения в виде .exe файла выложен на [GitHub](#)

Код разделён на 2 основных модуля: *main.py* и *Segmentation.py*. В модуле *Segmentation.py* находится класс, реализующий основную логику приложения.

```

import numpy as np
import cv2

class Segmentation:

    def __init__(self, path):
        self.path = path
        self.image = cv2.imread(path)

    self.modes_means = {'beginner': np.array([139.489, 99.937, 76.708]),
                        'intermediate1': np.array([73.192, 51.308, 64.041]),
                        'intermediate2': np.array([104.587, 91.975, 100.2])}

    self.thresholds = {'beginner': {1: np.array([[68, 78],
                                                [107, 125],
                                                [146, 170]]),
                                    2: np.array([[122, 32],
                                                [36, 42],
                                                [37, 51]]),
                                    3: np.array([[0, 5],
                                                [70, 110],
                                                [110, 182]]),
                                    4: np.array([[56, 69],
                                                [47, 68],
                                                [48, 49]]),
                                    5: np.array([[17, 30],
                                                [24, 34],
                                                [110, 129]]}),
                           }

```

Рис. 3: Модуль *Segmentation.py*

Модуль *main.py* создаёт графический интерфейс, импортирующий предыдущий модуль и реализующий весь функционал. После выбора изображения пользователь может его обработать, либо выбрать другое изображение.

```
|     QFileDialog, QLabel, QScrollArea,
|     QSsplitter
| )
from PyQt6.QtGui import QPixmap, QImage
from PyQt6.QtCore import Qt
from Segmentation import Segmentation
import numpy as np
import cv2

class CenteredButtonsApp(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Сегментация")

        self.layout = QVBoxLayout()

        self.image_label = QLabel()
        self.processed_image = QLabel()
        self.image_appeared = False
        self.image = None
        self.filename = None
        self.result_str = QLabel()
        self.result_str.setWordWrap(True)
        self.result_str.setText('')

        self.splitter = QSsplitter()
        # Create a QScrollArea
```

Рис. 4: Модуль *main.py*

5 Эксперименты

1. Эксперимент 1. Поиск треугольников через преобразование Хафа. Без должной пороговой бинаризации результаты получались неудовлетворительными.

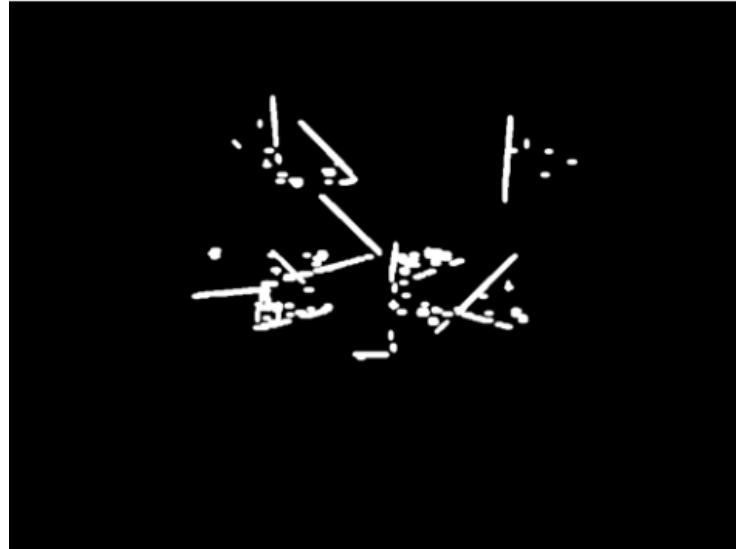


Рис. 5: Преобразование Хафа

2. Эксперимент 2. Сравнение контуров с фигураю треугольника путём функции близости двух контуров. Достаточно сильный метод, однако он требует должной бинаризации, так как находились не все треугольники

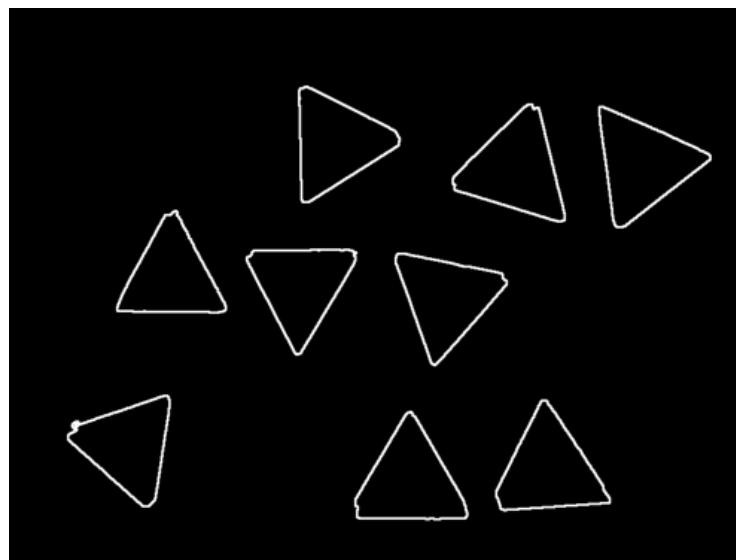


Рис. 6: Функция близости

3. Эксперимент 3. Использование различных методов выделения границ - ядра Собеля, Кирша, Приютта, фильтр Габора, преобразование Лапласа, бинаризация методом Оцу. Из всех проверенных методов лучшим оказался оператор Кэнни.

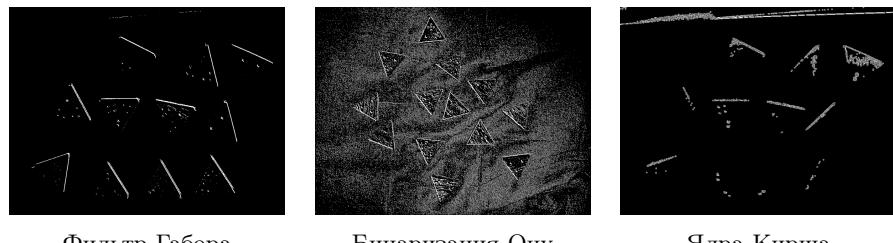


Рис. 7. Эксперимент 3

4. Эксперимент 4. Бинаризация путём гистограммного анализа через различные форматы представления изображения: rgb, lab, hsv, hls, ycbcr, cmyk, grayscale. Ниже пример гистограммы для канала l формата lab.

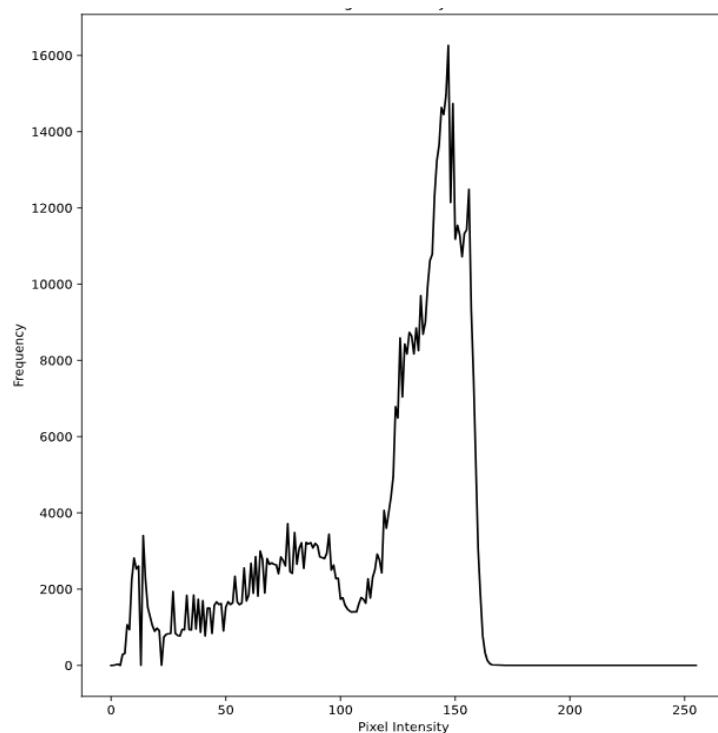


Рис. 8: Канал lightness

5. Эксперимент 5. Выделение пороговых значений для нахождения точек через различные форматы представления изображения. Применение кросс-маскировки между различными форматами. В итоге оставлен вариант с `rgb`.

6 Выводы

Были изучены и освоены методы обработки и сегментации изображений. Разработано и реализовано приложение для работы с изображениями фишек игрового набора «Тримино» уровня сложности Beginner и Intermediate. Реализованы точечные и пространственные преобразования для пороговой бинаризации и нахождения контуров. Изучены методы нахождения простых геометрических форм через функцию близости. Рассмотрены преобразования Хафа, Лапласа, Собеля, Кирша, Прюитта.