

Отчет по лабораторной работе №1

Изучение и освоение методов обработки и сегментации изображений

Каратыщев Дмитрий Иванович

Апрель 2024

1 Постановка задачи

Необходимо разработать и реализовать программу для работы с фотографиями «Кладбища самолётов», обеспечивающую:

1. Ввод и отображение на экране изображений;
2. Сегментацию изображений на основе точечных и пространственных преобразований;
3. Бинаризацию и подсчёт связных компонент, соответствующих отдельным объектам;
4. Оценку количества самолётов на выделенном фрагменте изображения.

Программа должна обеспечивать работу в режиме диалога: выбор файла с изображением, выполнение операций преобразования изображения, визуализацию этих операций и выдачу результатов подсчёта самолётов.

2 Описание данных

В качестве входных данных используются 3 цветных изображения «Кладбища самолётов», каждое из которых может быть изначально изменено путём выделения случайных прямоугольных участков.



Изображение 1

Изображение 2

Изображение 3

Рис. 1. Входные данные

3 Описание метода решения

Рассмотрим этапы обработки и сегментации входного изображения

1. Перевод изображения из формата **rgb** в формат **hsv** (*hue, saturation, value*). Этот шаг гораздо упрощает сегментацию, так как на одном из входных изображений присутствуют дороги с разным уровнем яркости, которые мешают выделять порог при бинаризации и находить конкретный цвет.
2. Выделение канала *hue* из hsv формата. Все самолёты имеют почти тот же самый цвет, что позволяет использовать канал *hue* для выделения диапазона оттенков, в которых находится каждый самолёт. Этот шаг преобразует цветное изображение в полутоновое.
3. Бинаризация изображения по двум порогам. Отбрасываются оттенки со значением ниже числа 90 и выше числа 127. Данные пороги были выбраны эмпирически, в основном, на основе Изображения 3, так как оно оказалось самым сложным для сегментации.
4. Выделение контуров на основе свёртки Кирша, которая использует 8 ядер размера 3×3 следующего вида:

$$\mathbf{g}^1 = \begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}, \mathbf{g}^2 = \begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}, \mathbf{g}^3 = \begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix}, \dots$$

Пиксель выходного изображения определяется как максимум из 8 значений, полученных применением каждого из приведённых ядер к окрестности пикселя. Данный этап помог убрать лишние объекты, оставшиеся после бинаризации, улучшить визуализацию результата, а также точнее отбирать связные компоненты для более корректного подсчёта числа самолётов в сравнении с обычной бинаризацией.

5. Использование *Spaghetti Labeling Algorithm* для выделения связных компонент на основе ближайших соседей каждого пикселя. Краткое описание алгоритма:

Алгоритм 1: Spaghetti Labeling Algorithm

```
1 Function SpaghettiLabeling(image):  
2   Инициализируем нулевую матрицу  $L$  с размерами image;  
3    $label \leftarrow 1$ ;  
4   for пиксель  $(i, j)$  в image do  
5     if пиксель  $(i, j)$  относится не к фону и не помечен  
6       then  
7         dfs( $i, j, label, image, L$ );  
8          $label \leftarrow label + 1$ ;  
9       end  
10    end  
11    return  $L$ ;  
12 Procedure dfs( $x, y, label, image, L$ ):  
13   Помечаем пиксель  $(x, y)$  меткой  $label$  в матрице  $L$ ;  
14   for каждый соседний пиксель  $(x', y')$  с  $(x, y)$  do  
15     if сосед  $(x', y')$  относится не к фону и не помечен then  
16       dfs( $x', y', label, image, L$ );  
17     end  
18   end
```

6. Визуализация результата сегментации и подсчёта числа самолётов.

4 Описание программной реализации

Для написания приложения использовался язык программирования Python с библиотеками PyQt6 для написания графического пользовательского интерфейса; scipy для использования 2D свёртки; numpy, opencv и PIL для сегментации изображений и поиска связанных компонент. Весь программный код и релиз приложения в виде .exe файла выложен на [GitHub](#)

Код разделён на 3 основных модуля: *main.py*, *Segmentation.py* и *ConnectedComponents.py*. В модуле *Segmentation.py* находится класс, реализующий бинаризацию изображения, перевод изображения из *rgb* в *hsv* и из *hsv* в *grayscale*, а также выделение краёв свёрткой Кирша

```

1 from PIL import Image
2 import numpy as np
3 from scipy import signal
4 import cv2
5
6
7 class Segmentation:
8
9     def __init__(self, paths):
10         self.paths = paths
11         self.images = self.rgb2hsv(self.read_images(paths))
12         self.gray_images = self.hsv2gray(self.images)
13
14         self.low_hue = 90
15         self.high_hue = 127
16
17         self.kirsch_kernels = np.array([[[[-3, -3, 5], [-3, 0, 5], [-3, -3, 5]],
18                                           [[-3, 5, 5], [-3, 0, 5], [-3, -3, -3]],
19                                           [[5, 5, 5], [-3, 0, -3], [-3, -3, -3]],
20                                           [[5, 5, -3], [5, 0, -3], [-3, -3, -3]],
21                                           [[5, -3, -3], [5, 0, -3], [5, -3, -3]],
22                                           [[-3, -3, -3], [5, 0, -3], [5, 5, -3]],
23                                           [[-3, -3, -3], [-3, 0, -3], [5, 5, 5]],
24                                           [[-3, -3, -3], [-3, 0, 5], [-3, 5, 5]]]])

```

Рис. 2: Модуль *Segmentation.py*

Модуль *ConnectedComponents.py* реализует подсчёт связанных компонент через функцию *connectedComponentsWithStats*

```

1 import numpy as np
2 import cv2
3
4
5 class ConnectedComponents:
6
7     def __init__(self, images):
8         self.images = images
9
10
11     def count_components(self, image, labels_returned=True):
12         num_labels, labels, _ = cv2.connectedComponentsWithStats(
13             image, connectivity=8)
14
15         if labels_returned:
16             return num_labels - 1, labels
17         else:
18             return num_labels - 1, None
19
20     def get_component_images(self):
21         component_images = dict()
22         num_components = []
23
24         for i in range(len(self.images)):
25             num_labels, labels = self.count_components(self.images[i])
26             num_components.append(num_labels)
27
28             component_image = np.zeros_like(self.images[i])
29
30             for label in range(1, num_labels + 1):
31                 component_mask = (labels == label).astype(np.uint8) * 255
32                 component_image = cv2.add(component_image, component_mask)
33
34             component_images[i] = component_image
35
36         return component_images, num_components

```

Рис. 3: Модуль *ConnectedComponents.py*

Модуль *main.py* создаёт графический интерфейс, импортирующий предыдущие два модуля и реализующий весь функционал. После выбора изображения пользователь может его обработать, выбрать другое, либо выполнить выделение случайного фрагмента изображения через кнопку «Вы-

делить случайный кадр». Выделение случайного фрагмента реализовано через `numpy.random.randint`. Интерфейс приложения приведён ниже.



Рис. 4: Модуль *ConnectedComponents.py*

5 Эксперименты

1. Эксперимент 1. Сегментация в *rgb* формате. Данный подход оказался не особо удачным, были опробованы разные варианты порогов, в основном выделялся канал синего цвета. Проблемным оказалось Изображение 3, оно оставляло много лишних артефактов, в то время как остальные изображения получались отличными.

2. Эксперимент 2. Выделение границ свёрткой с ядром
$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}.$$

Данная свёртка результировала в слишком большом числе связанных компонент, что отражало совсем не реалистичное число самолётов после бинаризации и в *rgb*, и в *hsv*.

3. Эксперимент 3. Использование различных ядер для увеличения контрастности, простого выделения границ и устранения шума. Не дало никаких улучшений.
4. Эксперимент 4. Перевод в *hsv*, свёртка Кирша по всему изображению, бинаризация на основе яркости и повторная свёртка Кирша. После первых двух шагов было замечено, что все самолёты стали достаточно яркими, что привело к идее бинаризации по яркости. Третье изображение было всё равно заполнено дорогами в итоге.
5. Эксперимент 5. Подбор порога для канала *hue*. Отбор всех возможных порогов проводился сразу на трёх изображениях, было замечено, что весь верхний диапазон относился к дорогам третьего изображения. Поэтому выбрана как верхняя (127), так и нижняя граница (90)

значений. Каждый порог оценивался итоговым числом самолётов и в итоге подобраны оптимальные значения. На данный момент полученные результаты числа самолётов следующие: 298 на первом, 252 на втором и 462 самолёта на 3-м изображении.

6 Выводы

Были изучены и освоены методы обработки и сегментации изображений. Реализованы точечное (бинаризация по порогу) и пространственное (2D свёртка) преобразования. Установлено, что сегментация изображений лишь на основе предложенных методов требует большое число экспериментов и эмпирических данных для корректных результатов, так как бинаризация может захватить лишние объекты, не относящиеся к искомым объектам. Эмпирическим путём получено, что качество сегментации может увеличиться, если переводить изображение в разные форматы представления пикселей (*rgb*, *hsv*, ...)