Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчёт по лабораторной работе №6 по дисциплине "Основы искусственного интеллекта"

Тема Компьютерное зрение

Студент Варламова Е. А.

Группа ИУ7-13М

Оценка (баллы) _____

Преподаватели Строганов Ю.В.

Москва — 2023 г.

СОДЕРЖАНИЕ

Введение

Компьютерное зрение – это раздел искусственного интеллекта, который ориентирован на изучение и разработку методов и технологий для анализа, обработки и интерпретации изображений и видео. Эта область науки позволяет компьютерам «видеть» и «понимать» содержимое изображений, подобно тому, как это делает человек.

Компьютерное зрение имеет широкий спектр практических применений, некоторые из них:

— медицинская диагностика: компьютерное зрение используется для анализа медицинских изображений, таких как рентгеновские снимки, снимки МРТ, ультразвуковые изображения и томография, с целью диагностики заболеваний и патологий;

— автоматизированное вождение: компьютерное зрение применяется для обнаружения и распознавания дорожных знаков, разметки дорог, определения пешеходов, других транспортных средств и объектов;

— промышленный контроль качества: компьютерное зрение используется для автоматической инспекции и контроля качества на производственных линиях.

Целью данной лабораторной работы является создание программы для обработки изображений с целью выявления ошибки отсутствия подписей у стрелочек при ветвлении на схеме алгоритмов.

Для этого необходимо решить следующие задачи:

— привести постановку задачи;

— провести обзор методов предварительной обработки изображений;

— провести обзор методов локализации и классификации объектов;

— провести обзор существующих решений для определения текста на изображении;

— разработать схему алгоритма обработки изображения для поиска ошибки;

— привести особенности реализации ПО, решающего поставленную задачу;

— провести исследование, состоящее в определении ошибки работы алгоритма.

1 | Аналитическая часть

1.1 Постановка задачи

Разработать ПО для обработки изображений с целью выявления ошибки отсутствия подписей у стрелочек при ветвлении на схеме алгоритмов.

1.2 Обзор методов предварительной обработки изображений

Существует несколько методов предварительной обработки изображений. Рассмотрим некоторые из них.

1. Подавление шумов, сглаживание. Одним из методов подавления шума и сглаживания является применение филтра Гаусса. Фильтр гаусса применяется для уменьшения визуального шума, размытия или сглаживания изображения, уменьшения резких переходов яркости между пикселями.

2. Преобразование цветов. Этот метод преобразует цветовую модель изображения из RGB (красный, зеленый, синий) в HSV (оттенок, насыщенность, значение). Это позволяет более удобно работать с оттенками, насыщенностью и яркостью цветов, что может быть полезно для коррекции цветового баланса, фильтрации по оттенку или насыщенности, а также для других операций, связанных с цветовыми характеристиками изображения.

3. Повышение резкости. Этот метод увеличивает резкость границ объектов на изображении, делая их более четкими и выразительными. Обычно это достигается путем увеличения контраста вдоль границ и усиления различий в яркости между соседними пикселями.

4. Повышение контрастности. Метод повышения контрастности используется для увеличения различий между яркими и темными областями изображения. Это делает изображение более выразительным, улучшает его визуальное восприятие и делает более четкими детали. Обычно контрастность

может быть изменена путем растяжения или сжатия гистограммы яркости изображения.

В данной работе для предварительной обработки изображения используются методы преобразования цветов и подавления шумов с помощью филтра Гаусса.

1.3 Обзор методов локализации и классификации объектов

1.3.1 Преобразование Хаффа

Преобразование Хаффа – это метод анализа изображений, который используется для обнаружения прямых линий или других форм. Он работает путем преобразования изображения в пространство параметров, где каждая точка преобразуется в набор параметров, определяющих прямую. Преимущества и недостатки данного метода:

Преимущества:

— Преобразование Хаффа позволяет обнаруживать прямые линии, даже если они имеют различные углы наклона или сдвиги.

— Можно применять для обнаружения групп прямых, что полезно при обнаружении объектов.

Недостатки:

— Преобразование Хаффа требует больших вычислительных ресурсов, особенно на больших изображениях.

— Не всегда эффективен для обнаружения кривых или нелинейных форм.

1.3.2 Контурный анализ

Контурный анализ – это метод выделения контуров объектов на изображении. Он основан на поиске перепадов яркости или цвета, т.е. линий, разделяющих области объектов на изображении. Преимущества и недостатки данного метода:

Преимущества:

— Контурный анализ работает отлично для обнаружения форм и контуров объектов различных форм и размеров.

— Можно применять для выделения деталей и текстур на изображении.

Недостатки:

— Может быть неэффективен в случае низкого качества изображения или наличия шумов.

— Не всегда способен правильно различать контуры объектов, особенно если они сильно перекрываются.

1.3.3 Сопоставление с шаблоном

Сопоставление с шаблоном – это метод сопоставления изображения с заранее известным образцом или шаблоном. Он используется для поиска совпадений между изображением и заданным шаблоном. Преимущества и недостатки данного метода:

Преимущества:

— Эффективен для обнаружения конкретных объектов или образцов на изображениях.

— Можно применять для распознавания образцов, таких как буквы, цифры и символы.

Недостатки:

— Точность сопоставления с шаблоном зависит от качества шаблона и совершенства совпадений.

— Может быть неэффективен в случае изменения масштаба, искажения или поворота объектов на изображениях.

Сравнивая вышеперечисленные методы, контурный анализ является наиболее предпочтительным. Он обладает хорошей способностью выделения контуров объектов различной формы, а также не зависит от точности сопоставления шаблонов или вычислительных ресурсов, требуемых для преобразования Хафа. Контурный анализ также более устойчив к шумам и способен работать с изображениями различного качества. Поэтому в данной работе для определения позиций блоков ветвления используется контурный анализ.

1.4 Обзор решений для распознавания текста на изображениях

Существует множество решений, способных распознавать текст на изображениях. Рассмотрим некоторые из них.

1. **FineReader** [1]
   - **Принцип работы:** FineReader использует технологии распознавания оптических знаков (OCR) для извлечения текста с изображений. Это включает в себя анализ формы символов, распознавание шаблонов и языковые модели.
   - **Преимущества:** Высокая точность распознавания, поддержка большого количества языков и форматов документов.
   - **Недостатки:** Проприетарная система, что может быть дорого и неудобно для некоторых пользователей. Требует установки на компьютер.
2. **Tesseract** [2]
   - **Принцип работы:** Tesseract - это программный пакет с открытым исходным кодом для распознавания текста, который использует нейронные сети для обучения моделей распознавания текста.
   - **Преимущества:** Бесплатное и с открытым исходным кодом, постоянно развивается благодаря сообществу разработчиков.
   - **Недостатки:** Низкая точность в сравнении с другими платными решениями, требует навыков программирования для настройки и использования.
3. **EasyOCR** [3]
   - **Принцип работы:** EasyOCR - бесплатная библиотека Python для распознавания текста на изображениях, которая использует методы глубокого обучения для распознавания текста на изображениях.
   - **Преимущества:** Простота использования, поддержка большого количества языков, высокая скорость обработки.
   - **Недостатки:** Может быть менее точной по сравнению с некоторыми другими платными решениями, поскольку основана на моделях глубокого обучения.

Использование EasyOCR является наиболее предпочтительным, так как решение обладает высокой скоростью обработки изображений и точностью распознавания текста относительно других решений, кроме того, поддерживает большинство языков программирования.

Вывод

В данном разделе была приведена постановка задачи, проведён обзор методов предварительной обработки изображений и методов локализации и классификации объектов. Было принято решение для определения позиций блоков ветвления использовать контурный анализ, для предварительной обработки изображений – подавление шума и преобразование цветов, а для распознавания текста на изображениях – решение EasyOCR.

2 | Конструкторская часть

2.1 Схема алгоритма

В соотвествии с заданием, необходимо реализовтаь алгоритм, который по входному изображению выводит сообщение об ошибке или об ее отсутствии. Соответствующий алгоритм приведен на рисунке 2.1.

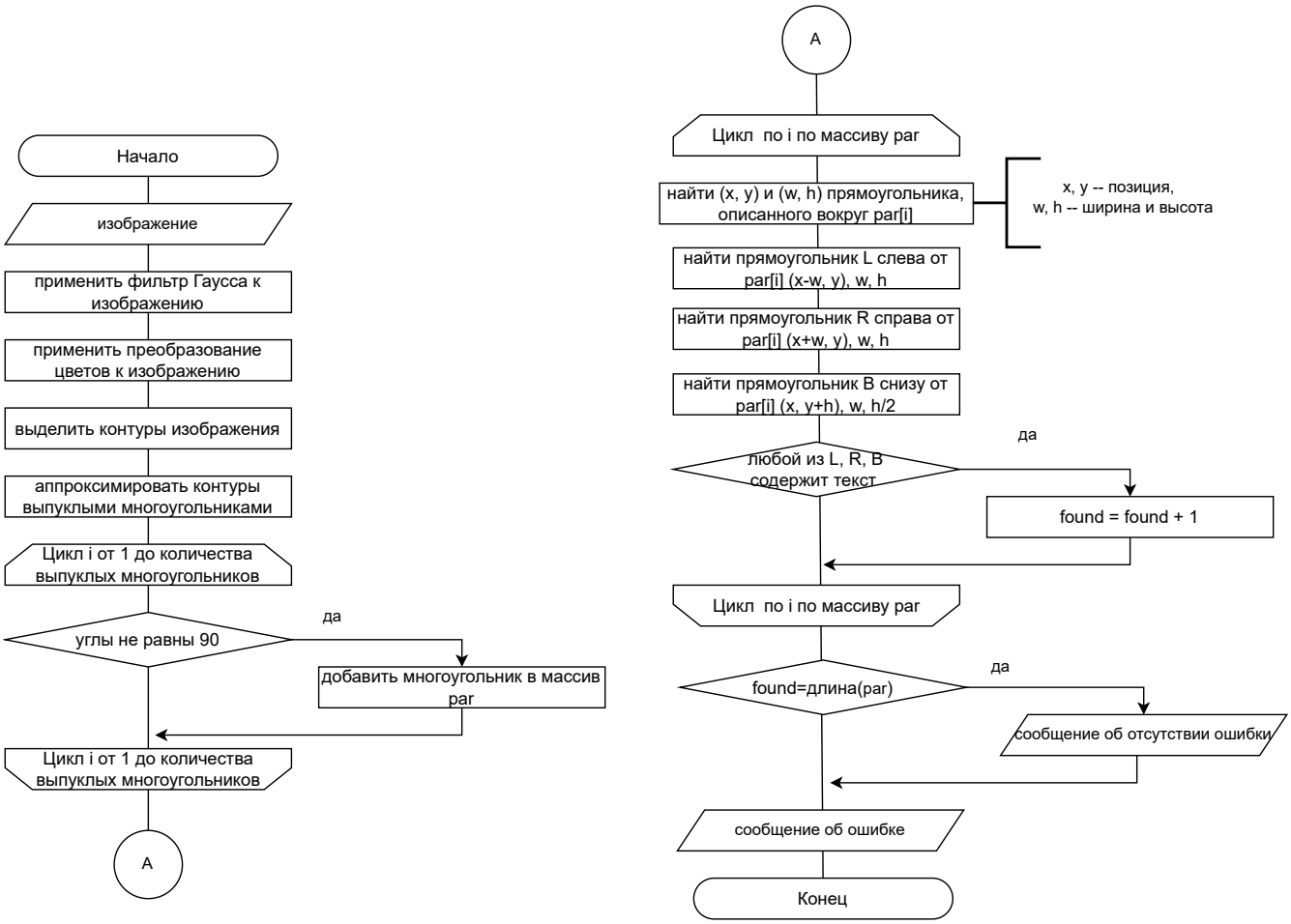

Рис. 2.1: Схема алгоритма решения задачи

Вывод

В данном разделе была приведена схема алгоритма, решающего поставленную задачу.

3 | Технологическая часть

3.1 Выбор средств разработки

В качестве языка программирования был использован язык Python, поскольку этот язык кроссплатформенный и для него разработано огромное количество библиотек и модулей, решающих разнообразные задачи.

В частности, имеются библиотеки, включающие в себя инструменты для обработки изображений. В данной работе была использована библиотека «opencv» [4] для обработки изображений, а также библиотека «easyocr» [3] для распознавания текста на изображении.

3.2 Листинги ПО

В листинге 3.1 представлена реализация алгоритма решения поставленной задачи.

Листинг 3.1: Реализация алгоритма решения задачи

```
import cv2
import numpy as np
import os
import easyocr
def DrawContours(image, contours, name):
    cont = image.copy()
    for contour in contours:
        cv2.drawContours(cont, [contour], 0, (0, 255, 0), 3)
    cv2.imwrite(name, cont)
def FindContours(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150)
    contours, _ = cv2.findContours(edges, cv2.RETR_TREE, cv2.
       CHAIN_APPROX_SIMPLE)
    return contours
def Find4AngleContours(image):
    contours = FindContours(image)
    angle4_contours = []
    for contour in contours:
        epsilon = 0.01 * cv2.arcLength(contour, True)
```

```
20          approx = cv2.approxPolyDP(contour, epsilon, True)
21          area = cv2.contourArea(approx)
22          image_area = image.shape[0] * image.shape[1]
23          if len(approx) == 4 and area > 1e-3 * image_area:
24              angle4_contours.append(approx)
25      return angle4_contours
26
27  def FindIfContours(image):
28      angle4_contours = Find4AngleContours(image)
29      parallelograms = []
30      for approx in angle4_contours:
31          check_90 = False
32          for i in range(len(approx) - 1):
33              vector1 = np.reshape(approx[i] - approx[i - 1], (2))
34              vector2 = np.reshape(approx[i + 1] - approx[i], (2))
35              cosine_angle = np.dot(vector1, vector2) / (np.linalg.norm(
                    vector1) * np.linalg.norm(vector2))
36              if abs(cosine_angle) < 0.1 :
37                  check_90 = True
38                  break
39          if not check_90:
40              parallelograms.append(approx)
41      return parallelograms
42  def AnalyzeText(image, parallelograms):
43      found = False
44      for r in parallelograms:
45          x, y, w, h = cv2.boundingRect(r)
46          to_test_areas = []
47          to_test_areas.append(image[y:y+h, x+w:x+2*w]) # right
48          to_test_areas.append(image[y:y+h, x-w:x]) # left
49          to_test_areas.append(image[y+h:y+h+int(h/2), x:x+w]) # bottom
50          for roi in to_test_areas:
51              if roi.any():
52                  gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
53
54                  reader = easyocr.Reader(['ru', 'en'])
55                  text = reader.readtext(gray, detail=0, paragraph=True)
56                  if '    ' in text or '    ' in text or '      ' in text or '
                            ' in text:
57                      found = True
58      if len(parallelograms) != 0 and not found:
59          return True
60      return False
61  import sys
62  image_name = sys.argv[1]
63  image = cv2.imread(image_name)
64  parallelograms = FindIfContours(image)
65  error_found = AnalyzeText(image, parallelograms)
66  print(error_found)
```

Вывод

В данном разделе были обоснованы средства реализации ПО и реализовано ПО, решающее поставленную задачу.

4 | Исследовательская часть

4.1 Определение ошибки работы алгоритма

Цель: провести исследование, состоящее в определении ошибки работы алгоритма.

Для определения ошибки использовался набор изображений схем алгоритмов из 95 изображений.

Результат Было выяснено, что алгоритм работает с точностью 91% (алгоритм дал 91% верных ответов).

Вывод

В данном разделе было проведено исследование, состоящее в определении ошибки работы алгоритма. Было выяснено, что алгоритм работает с точностью 91% (алгоритм дал 91% верных ответов).

ЗАКЛЮЧЕНИЕ

Целью данной лабораторной работы являлось создание программы для обработки изображений с целью выявления ошибки отсутствия подписей у стрелочек при ветвлении на схеме алгоритмов. Цель работы была достигнута Для этого было необходимо решить следующие задачи:

— привести постановку задачи;

— провести обзор методов предварительной обработки изображений;

— провести обзор методов локализации и классификации объектов;

— провести обзор существующих решений для определения текста на изображении;

— разработать схему алгоритма обработки изображения для поиска ошибки;

— привести особенности реализации ПО, решающего поставленную задачу;

— провести исследование, состоящее в определении ошибки работы алгоритма.

.

В результате исследования было выяснено, что разработанный алгоритм работает с точностью 91% (алгоритм дал 91% верных ответов).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Библиотека распознавания текста на изображениях finereader [Электронный ресурс]. — Режим доступа: URL: https://www.abbyy.com/en-gb/finereader (дата обращения: 19.12.2023).
2. Библиотека распознавания текста на изображениях tesseract [Электронный ресурс]. — Режим доступа: URL: https://github.com/tesseract-ocr/tesseract (дата обращения: 19.12.2023).
3. Библиотека распознавания текста на изображениях EasyOCR [Электронный ресурс]. — Режим доступа: URL: https://github.com/JaidedAI/EasyOCR (дата обращения: 19.12.2023).
4. Библиотека обработки изображений opencv [Электронный ресурс]. — Режим доступа: URL: https://opencv.org (дата обращения: 19.12.2023).