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»

Лабораторная работа № 6 по дисциплине «Основы искусственного интеллекта»

Тема Компьютерное зрение

Студент Сапожков А.М.,

Группа ИУ7-13М

Преподаватель Строганов Ю.В.

Москва, 2024

Содержание

ВВЕДЕНИЕ

Кластеризация является одной из важнейших задач в области анализа данных. Она используется для группировки объектов в такие подмножества (кластеры), в которых объекты внутри каждого кластера максимально схожи, а объекты из разных кластеров максимально различны. Задача кластеризации широко применяется в различных областях, включая обработку текстов, анализ изображений, биоинформатику и многие другие.

Целью данной лабораторной работы является разработка программы для определения нарушения интервала после и до заголовка раздела (подраздела) с помощью средств компьютерного зрения.

Задачи данной лабораторной работы:

1) выбрать программный инструмент для анализа изображений;
2) разработать и реализовать алгоритм проверки нарушения интервала после и до заголовка раздела (подраздела на основе средств компьютерного зрения;
3) протестировать реализованный анализатор на реальных отчётах студентов.

1 Аналитическая часть

1.1 Постановка задачи

Электронные отчёты по ЛР студенто представляют собой многокомпонентные документы, состоящие из множества элементов (титульный лист, содержание, основная часть, приложения и т.д.). Для успешной сдачи отчёта необходимо его строгое соответствие установленным требованиям по

— структуре документа (наличие разделов, их последовательность);
— форматированию (шрифты, размеры текста, отступы);
— содержанию (корректность данных, орфография);
— визуальной компоновке (поля, расположение заголовков, читабельность).

Однако, студенты часто допускают различные ошибки: неверное оформление титульного листа, отсутствие обязательных разделов, неправильно оформленные таблицы, орфографические или структурные ошибки.

Для автоматизации проверки отчётов предлагается разработать программный комплекс для выявления ошибок в отчётах. Основное внимание уделяется обработке сканов или скриншотов электронных отчётов, что по сути является задачей анализа изображения, включающей в себя

— распознавание содержимого документа;
— выделение ключевых элементов (заголовки, текстовые блоки, таблицы);
— поиск ошибок, основанный как на структурных, так и на содержательных требованиях.

2 Конструкторская часть

На рисунке 2.1 представлен алгоритм проверки отступов заголовков.

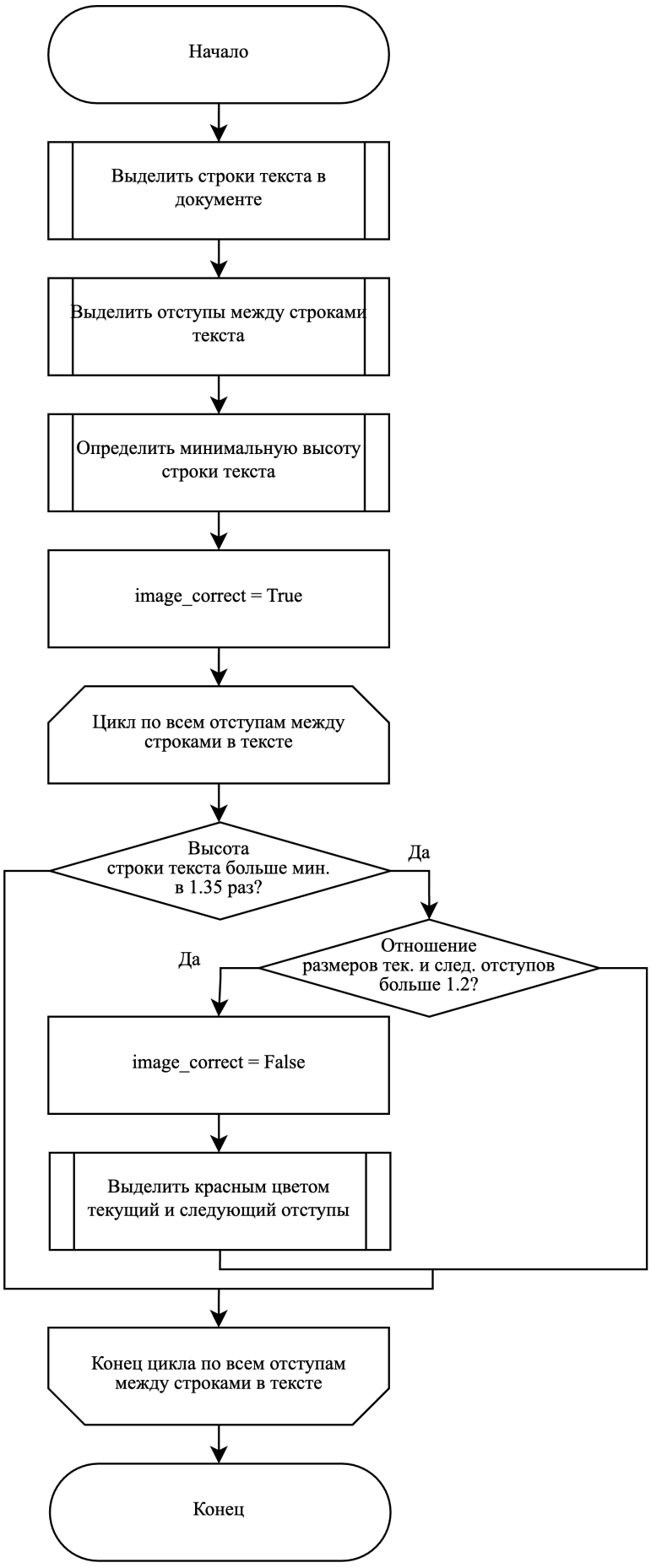


Рисунок 2.1 — Схема алгоритма проверки отступов заголовков

3 Технологическая часть

3.1 Средства реализации

Для автоматизации анализа электронных отчётов первым шагом является обработка самого изображения (фотографии или скана). Поскольку отчёты могут быть сканированы под углом, с тенями, шумами или недостаточной читаемостью текста, необходимо предусмотреть возможность

— устранения искажений изображений (перспективных, геометрических), удаления шумов, коррекции яркости и контраста;

— разделения документов на области;

— распознавания текста в документах;

— проверки соответствия визуальных элементов документов заданным шаблонам.

Для выполнения описанных задач наиболее актуальным инструментом является библиотека OpenCV [2] для языка программирования Python [1], которую предлагается использовать в данной работе.

3.2 Реализация алгоритма

На листинге 3.1 представлена реализация программы для определения нарушения интервала после и до заголовка раздела (подраздела).

Листинг 3.1 — Сравнение алгоритмов кластеризации K-Means и K-Medoids

```
import cv2

filename = "neg2.png"
relative_indent_threshold = 1.2
relative_text_height_threshold = 1.35

img = cv2.imread(filename)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, thresh1 = cv2.threshold(gray, 0, 255, cv2.THRESH_OTSU | cv2.
   THRESH_BINARY_INV)

# Specify structure shape and kernel size.
# Kernel size increases or decreases the area of the rectangle to be
   detected.
# A smaller value like (10, 10) will detect each word instead of a
   sentence.
rect_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (1, 1))

dilation = cv2.dilate(thresh1, rect_kernel, iterations = 1)
```

```
contours, hierarchy = cv2.findContours(dilation, cv2.RETR_EXTERNAL, cv2
    .CHAIN_APPROX_NONE)
res = img.copy()
rects = [cv2.boundingRect(cnt) for cnt in contours]

class ContourComparator(tuple):
  def __lt__(self, other):
    return self[1] < other[1] or self[0] < other[0]

rects.sort(key=ContourComparator)

i = 0
while i < len(rects) - 1:
  (x, y, w, h) = rects[i]
  (x_next, y_next, w_next, h_next) = rects[i+1]
  if y_next <= y + h:
    x_new = min(x, x_next)
    y_new = min(y, y_next)
    w_new = max(x + w, x_next + w_next) - x_new
    h_new = max(y + h, y_next + h_next) - y_new
    rects[i] = (x_new, y_new, w_new, h_new)
    del rects[i+1]
  else:
    i += 1

rects = [(x, y, w, h) for (x, y, w, h) in rects if w / h > 1]
spaces = [(
  5,
  rects[i][1] + rects[i][3],
  img.shape[1] - 10,
  rects[i+1][1] - rects[i][1] - rects[i][3],
) for i in range(len(rects) - 1)]

h_min = rects[0][3]
for rect in rects:
  print(rect)
  (x, y, w, h) = rect
  if h < h_min:
    h_min = h
  # cv2.rectangle(res, (x, y), (x + w, y + h), (0, 255, 0), 2)
```

```
print(f"min h = {h_min}\n")

image_correct = True
for i in range(len(spaces) - 1):
  print(spaces[i])
  (x, y, w, h) = spaces[i]
  (x_next, y_next, w_next, h_next) = spaces[i+1]
  text_height = rects[i+1][3]
  #cv2.rectangle(res, (x, y), (x + w, y + h), (255, 0, 0), 2)
  if text_height/h_min > relative_text_height_threshold and \
    max(h, h_next) / min(h, h_next) >= relative_indent_threshold:
    image_correct = False
    cv2.rectangle(res, (x, y), (x + w, y + h), (0, 0, 255), 2)
    cv2.rectangle(res, (x_next, y_next), (x_next + w_next, y_next +
       h_next), (0, 0, 255), 2)

if not image_correct:
  cv2.imwrite("recognized_" + filename, res)
  print(f"image has errors, see recognized_{filename}")
else:
  print("image has no errors")
```

4 Исследовательская часть

В данном разделе будет описано тестирование реализованного анализатора изображений.

4.1 Образец отчёта без ошибок

На рисунке 4.1 представлен скриншот электронного отчёта, не содержащий нарушений интервала после и до заголовка раздела (подраздела).

Реализованный анализатор не выявил нарушений.

4.2 Образец отчёта без ошибок

На рисунках 4.2 и 4.3 представлены скриншоты электронных отчётов, содержащие нарушения интервала после и до заголовка раздела (подраздела).

Выявленные нарушения анализатор выделил красным цветом.

Вывод

Реализованный анализатор успешно прошёл тесты на реальных отчётах студентов. Тот факт, что в коде программы не используются абсолютные величины (высоты текста, отступов и т.д.), позволяет не предъявлять требований к разрешению анализируемых изображений.

IM3: определение результирующей функции принадлежности $\mu_{res}(y)$ выходного значения (аккумуляция) с использованием оператора MAX. [1]

Блок «**ДЕФАЗЗИФИКАЦИЯ**» **(DEFUZZIFICATION)** на основе результирующей функции принадлежности $\mu_{res}(y)$ вычисляет чёткое числовое значение y^* выходного параметра, являющееся результатом для входных числовых значений x_1^*, x_2^*. Данная операция выполняется посредством **механизма дефаззификации**, который определяет метод вычисления. [1]

1.2 Методы дефаззификации

Наиболее известными методами дефаззификации являются

— метод среднего максимума (Middle of Maxima, MM);

— метод первого максимума (First of Maxima, FM);

— метод последнего максимума (Last of Maxima, LM);

— метод центра тяжести (Center of Gravity, CG);

— метод центра сумм (Center of Sums, CS);

— метод высот (Height, H). [1]

Далее перечисленные будут рассмотрены более подробно метод среднего максимума и метод центра тяжести. [1]

Метод среднего максимума (MM). Функцию принадлежности можно рассматривать как функцию, которая представляет информацию о сходстве между отдельными элементами множества и о наиболее типичном его элементе. С учётом функции принадлежности, соответствующей «среднему» значению роста, человек, имеющий рост 170 см, является типичным представителем данной категории роста (степень принадлежности равна 1), в то время как человека, имеющего рост 175 см, можно со степенью0.5 охарактеризовать как «среднего роста» и со степенью 0.5 — как «высокого». Иными словами, он частично соответствует как людям среднего роста, так и высоким людям. Таким образом, можно положить, что наиболее типичным представителем нечёткого множества B^*, полученного в результате вывода и задаваемого функцией принадлежности $\mu_{B^*}(y) = \mu_{res}(y)$, является значение y^*, имеющее максимальную степень принадлежности. Следует отметить, что множество таких значений часто может содержать более одного элемента и даже бесконечное число элементов. Решением в данной ситуации будет представление результирующего множества средним значением, получаемым по формуле

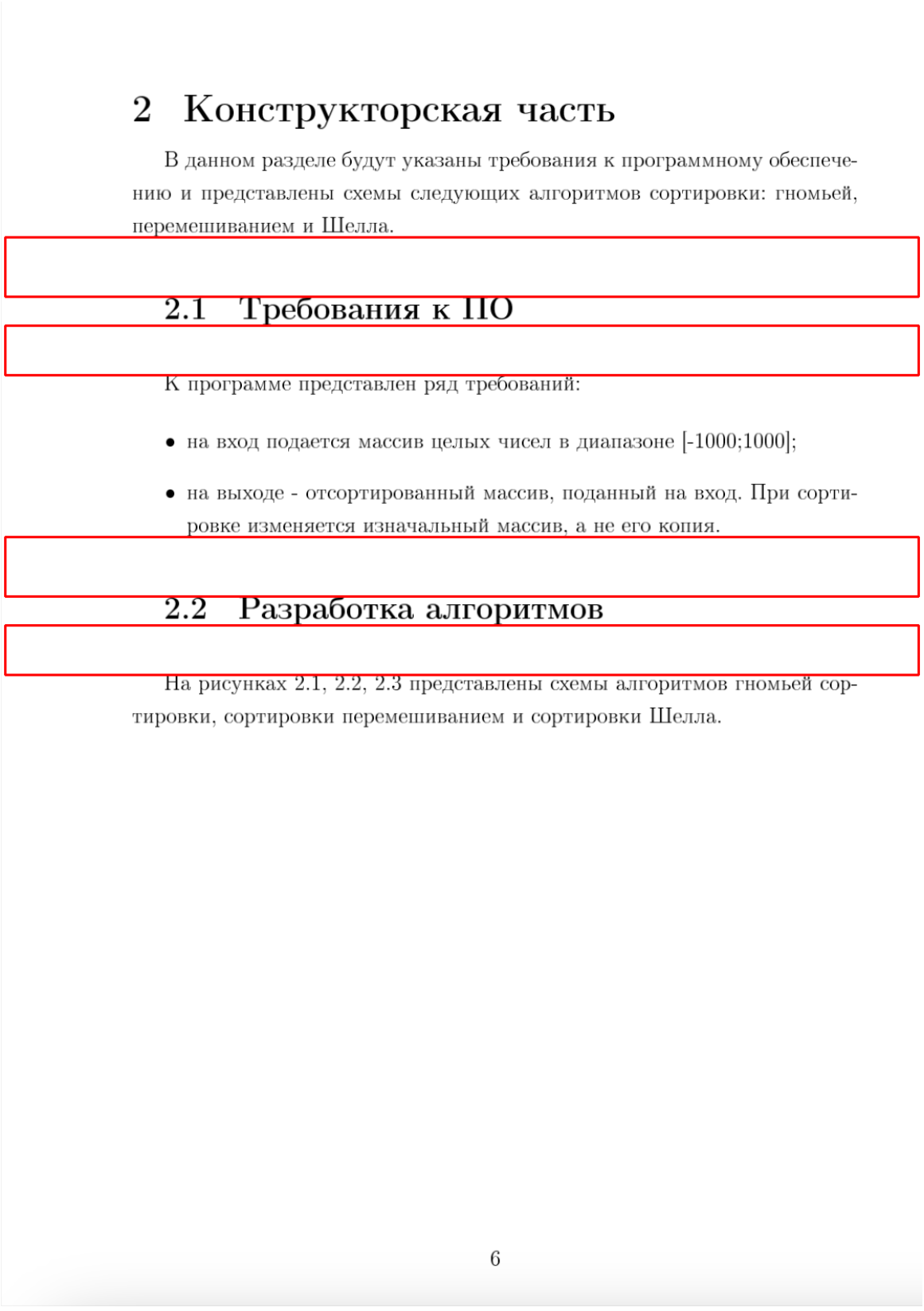
$$y^* = 0.5 \cdot (y_1^* + y_2^*). \tag{1.1}$$

Именно поэтому рассмотренный метод назван методом среднего максимума. [1]

Достоинством метода MM является простота вычислений, что допускает использование в системах управления более дешёвых микропроцессоров. Вместе с тем, простота вычислений достигается ценой определённых недостатков. [1]

Недостаток метода MM состоит в том, что на результат дефаззификации влияет только нечёткое множество Bj , имеющее наибольшую степень активизации — множества, активизи-

Рисунок 4.1 — Образец отчёта без ошибок



2 Конструкторская часть

В данном разделе будут указаны требования к программному обеспечению и представлены схемы следующих алгоритмов сортировки: гномьей, перемешиванием и Шелла.

2.1 Требования к ПО

К программе представлен ряд требований:

- на вход подается массив целых чисел в диапазоне [-1000;1000];
- на выходе - отсортированный массив, поданный на вход. При сортировке изменяется изначальный массив, а не его копия.

2.2 Разработка алгоритмов

На рисунках 2.1, 2.2, 2.3 представлены схемы алгоритмов гномьей сортировки, сортировки перемешиванием и сортировки Шелла.

6

Рисунок 4.2 — Образец отчёта с ошибками (1)

2 Конструкторская часть

В данном разделе будут рассмотрены схемы алгоритмов сортировок (битонная, блочная и расческой), а также найдена их трудоемкость.

Требования к программному обеспечению

Ряд требований к программе:

- на вход подается массив целых чисел в диапазоне от -10000 до 10000;
- возвращается отсортированный по месту массив, который был задан в предыдущем пункте.

2.1 Разработка алгоритмов

На рисунках 2.1, 2.2 и 2.3 представлены схемы алгоритмов сортировки – битонная, блочная и расческой.

7

Рисунок 4.3 — Образец отчёта с ошибками (2)

ЗАКЛЮЧЕНИЕ

В рамках лабораторной работы была разработана программа для определения нарушения интервала после и до заголовка раздела (подраздела) с помощью средств компьютерного зрения. Все поставленные задачи были выполнены.

1) выбран программный инструмент для анализа изображений;
2) разработан и реализован алгоритм проверки нарушения интервала после и до заголовка раздела (подраздела на основе средств компьютерного зрения;
3) реализованный анализатор протестирован на реальных отчётах студентов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Python [Электронный ресурс]. — Режим доступа, URL: https://www.python.org/ (дата обращения: 26.12.2024).=

2. OpenCV-Python Tutorials [Электронный ресурс]. — Режим доступа, URL:https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html (дата обращения: 26.12.2024).