МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Костромской государственный университет» (КГУ)

Институт автоматизированных систем и технологий

Кафедра информационных систем и технологий

Направление подготовки/Специальность\* 09.03.02

Информационные системы и технологии

Профиль Разработка программного обеспечения информационных систем

Дисциплина Технологии компьютерного зрения

ОТЧЁТ

Лабораторная работа №2: Предварительная обработка изображений.

Выполнили студенты Смирнов Кирилл Андреевич

Шкунов Владимир Викторович

Группа 22-ИСбо-2а

Проверил Орлов Александр Валерьевич

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_

Кострома 2024

**Задание**

Задание 1: Создайте приложение на Python, которое загружает изображение из заданного файла, преобразует в оттенки серого, а затем выполняет прямое пороговое преобразование. Имя файла и величина порога должны вводиться с клавиатуры перед загрузкой файла, либо браться из параметров командной строки.

Результат преобразования должен быть показан в окне.

Для преобразования изображения в оттенки серого используйте функцию cv2.cvtColor() в режиме cv2.COLOR\_BGR2GRAY, а для порогового преобразования - cv2.threshold(). Режим преобразования - cv2.THRESH\_BINARY.

Задание 2: Модифицируйте программу так, чтобы изначально введённое пороговое значение менялось по нажатию на клавиши стрелок "вверх" и "вниз", а закрытие окна выполнялось по Esc. Для этого анализируйте значение, возвращаемое cv2.waitKeyEx().

Если начальное значение порога не указано, или указано некорректно, используйте метод Оцу (cv2.THRESH\_OTSU) и выберите значение порога по возвращаемому значению cv2.threshold().

По закрытию окна программа должна вывести последнее значение порога.

Задание 3: Модифицируйте программу из задания 1 так, чтобы она использовала алгоритм адаптивного порогового преобразования. Параметры алгоритма (константа и размер окна) должны указываться пользователем, с клавиатуры или в командной строке.

Задание 4: Модифицируйте вашу программу из задания 3, чтобы параметры алгоритма изменялись клавишами стрелок (вверх/вниз - константа, вправо/влево - размер окна). Начальные значения вводятся с клавиатуры.

Используя созданное приложение, подберите параметры преобразования так, чтобы текст на изображении sonnet.png был как можно более читаем.

После каждого изменения параметров программа должна вывести их текущие значения.

Дополнительное 1: Измените программу из задания 2 следующим образом. Загрузите в качестве исходного изображения файл contrast.png. Используйте функцию cv2.cvtColor(), чтобы преобразовать это изображение в цветовую систему HSV (hue, saturation, value - оттенок, насыщенность, интенсивность). Выполняйте пороговое преобразование по каналу 1 (насыщенность) аналогично заданию 2.

Как и в задании 2, начальный выбор порога должен выбираться с помощью алгоритма Оцу, а клавиши "вверх" и "вниз" должны увеличивать или уменьшать этот порог в диапазоне 0..255.

Дополнительное 2: Измените программу из дополнительного задания 1 следующим образом.

Получив бинарное изображение binary, постройте на его основе маску цветных объектов (binary > 0) и маску серых объектов (binary == 0). Используя эти маски, залейте серым (127, 127, 127) "цветные" пиксели на одной копии исходного изображения и "серые" пиксели на другой копии.

**Вопросы**

1. В какие условиях алгоритм Оцу работает лучше всего?
2. Какие достоинства и недостатки у алгоритма адаптивного преобразования?
3. В чем разница между параметром “порог” прямого преобразования и параметров “константа” адаптивного?

**Вопрос №1**

Алгоритм ОЦУ эффективен , когда изображение имеет хорошо отделенные от фота объекты, низкий уровень шума и нормальное освещение. Он справляется с задачами бинаризации изображения, если фон и объекты разделяются по яркости. Алгоритм может автоматически определить оптимальный порог бинаризации на основе гистограммы яркости пикселей.

**Вопрос №2**

Достоинства:

1. Так как алгоритм разделяет изображение на небольшие блоки и применяет пороговое преобразование независимо к каждому блоку. Это позволяет эффективно распределить вычисление, что важно для обработки больших изображений.
2. Алгоритм может быть более эффективным при работе с изображениями, где освещенность неравномерна и неоднородна.

Недостатки:

1. Не может выполняться “на месте”
2. Плохо работает на объектах большой (>N) толщины

**Вопрос №3**

Параметр “порог”: обычно представляет значение яркости или цвета, которое использовано для разделения объектов и фона на изображении. Он фиксирован и применяется ко всему изображению для бинаризации.

Параметр “константа” адаптивного: определяет метод адаптивной бинаризации, размер окрестности или весовые коэффициенты. Каждый пиксель рассматривается индивидуально и порог бинаризации рассчитывается на основе окрестности пикселя с использованием этой константы.

Вывод: “порог” является фиксированным значением для всего изображения, а “константа” влияет на алгоритм бинаризации путем адаптации порога к окружающим пикселям для каждой части изображения.

**Скриншоты работы**

**Задание 1**

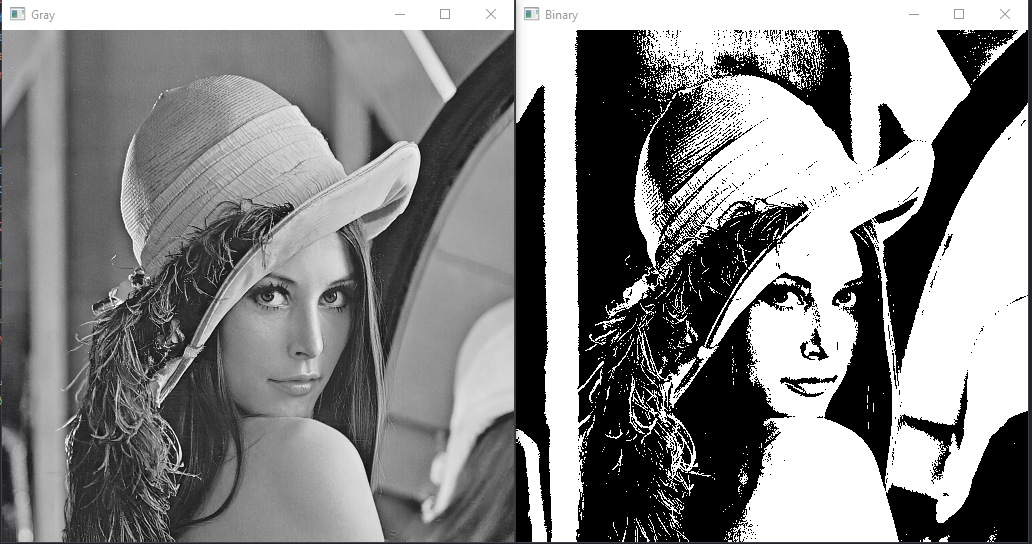
****

Рис. 1. Вывод первого задания, слева преобразование в оттенки серого, справа изображение с порогом 127.

**Задание 2**

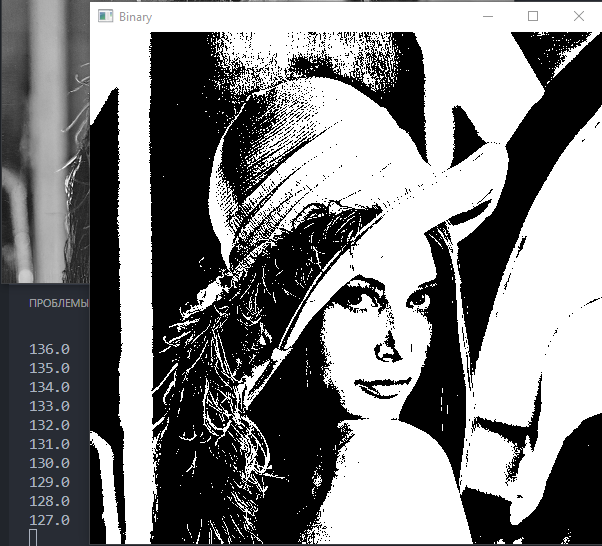
****

Рис. 2. Вывод второго задания.

**Задание 3**

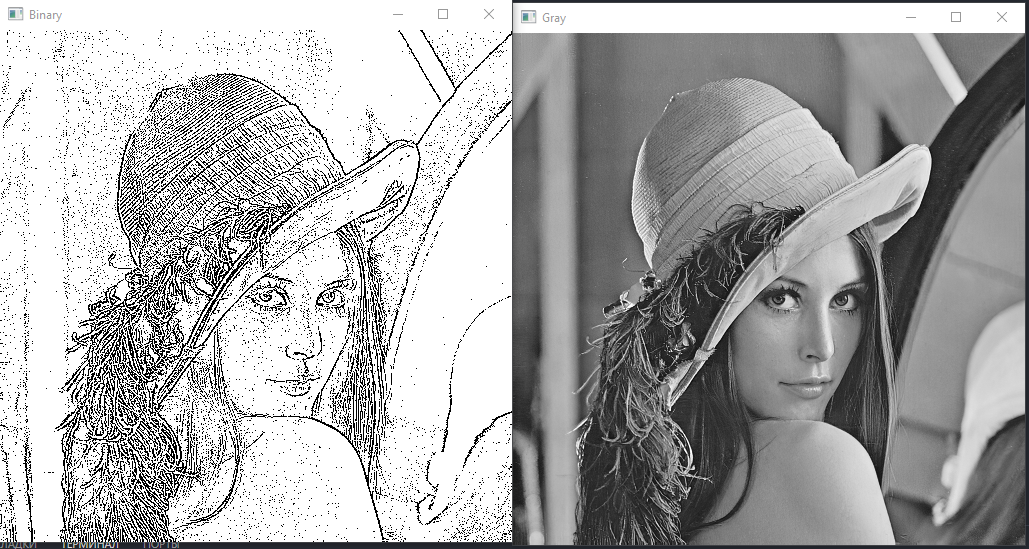
****

Рис. 3. Вывод третьего задания. Размер 5, константа 5.

**Задание 4**

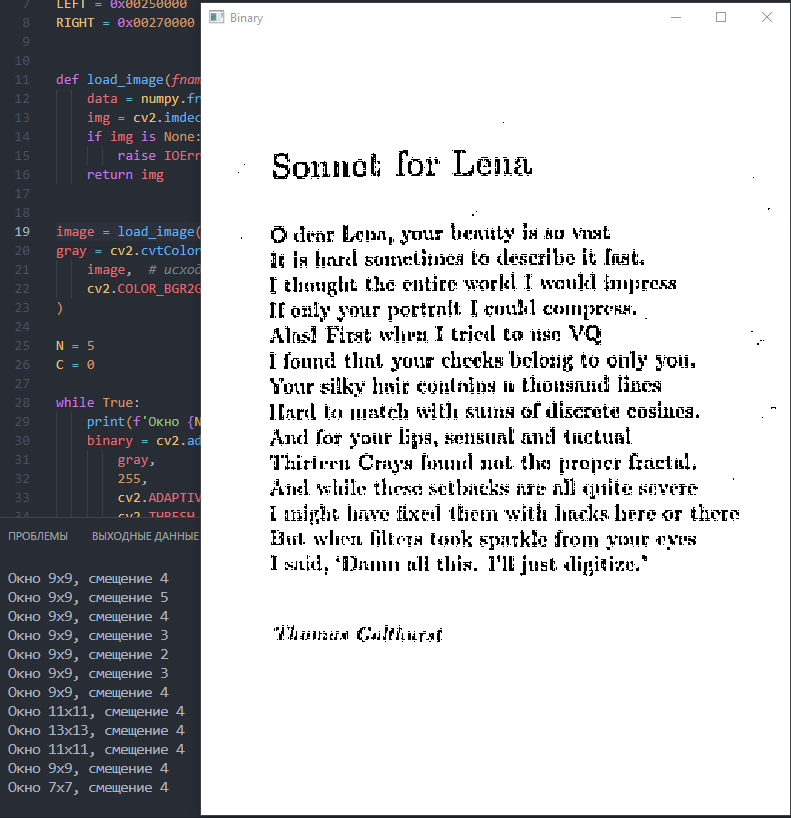
****

Рис. 4. Вывод четвертого задания.

**Дополнительно задание 1**

****

Рис. 5. Вывод первого дополнительного задания при минимальном пороговом значении.

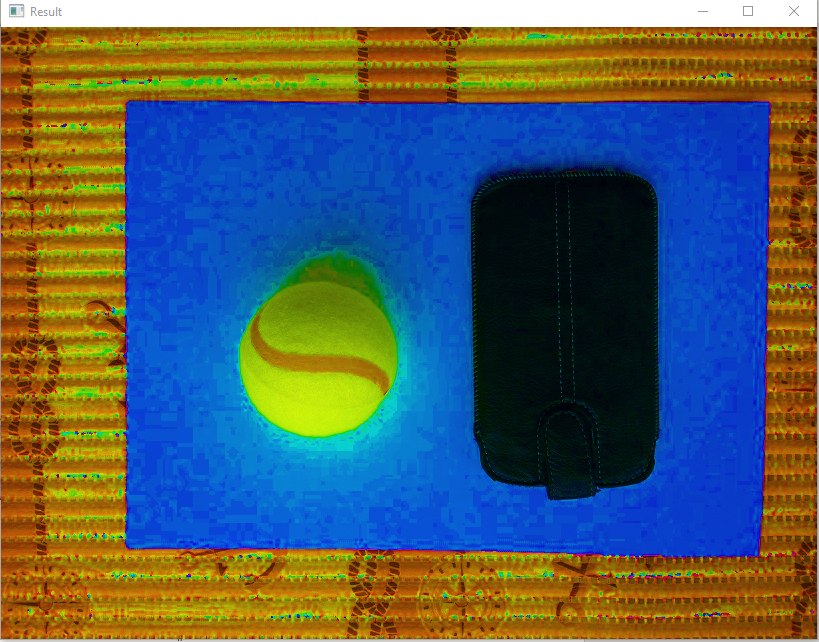


Рис. 6. Вывод первого дополнительного задания при максимальном пороговом значении.

**Дополнительно задание 2**

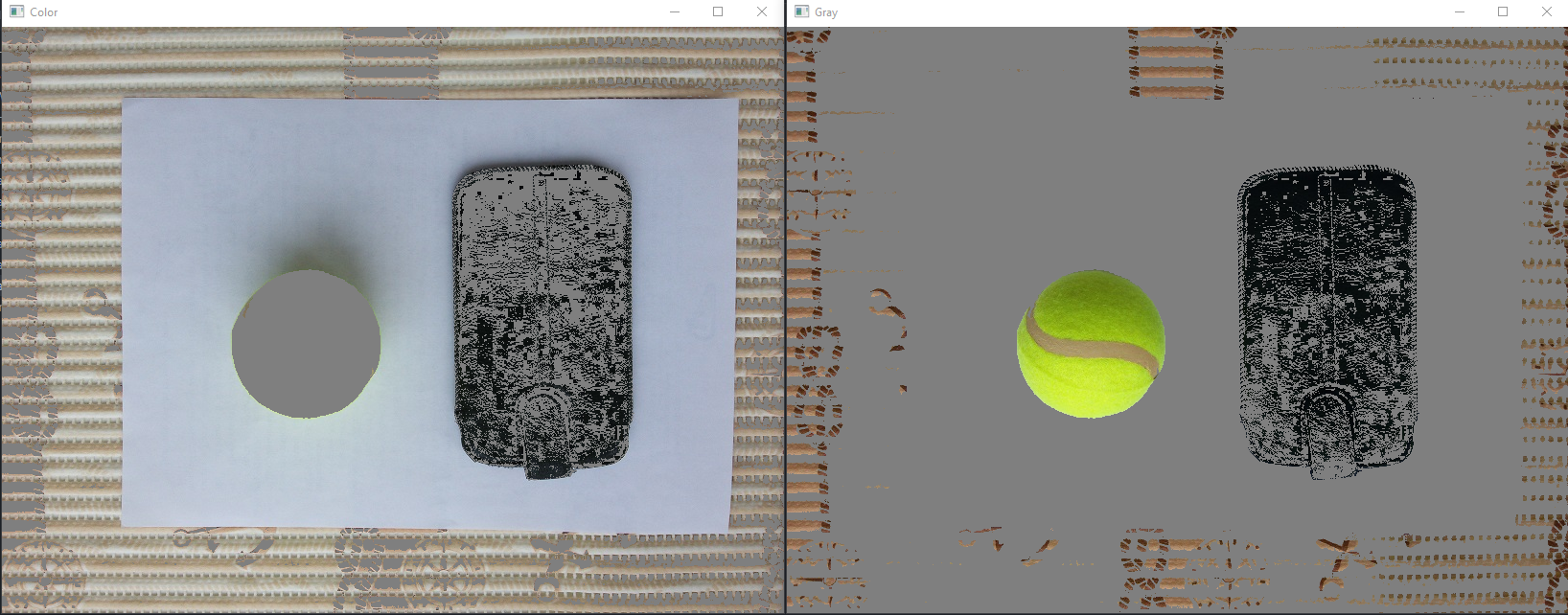
****

Рис. 7. Вывод второго дополнительного задания. Слева заливка цветных пикселей серым, справа заливка серых объектов серым.