МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Костромской государственный университет»

(КГУ)

ИАСТ

Кафедра автоматизированных систем и технологий

09.03.02

Направление подготовки/Специальность Информационные системы и технологии

Дисциплина Технологии компьютерного зрения

# Лабораторная №2.

# Предварительная обработка изображений.

Выполнили студенты Копосов Лев Владимирович

Копосов Владимир Владимирович

Группа 22-ИСбо-1б

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

Кострома

Вопросы:

1. В каких условиях алгоритм Оцу работает лучше всего?

Алгоритм Оцу работает лучше всего при условиях:

Так как алгоритм Оцу определяет порог путем анализа распределения яркостей пикселей, то лучшая работа алгоритма осуществляется при равномерном освещении изображения.

Алгоритм Оцу чувствителен к шуму, значит изображение должно иметь минимальный шум. Размытие изображения может повлиять на качество работы алгоритма, так как значение порога адаптировано к входному изображению.

Лучшая работа алгоритма Оцу демонстрируется на черно-белом изображении.

1. Какие достоинства и недостатки у алгоритма адаптивного преобразования?

У алгоритма адаптивного преобразования имеются и достоинства, и недостатки:

достоинства:

1. сложнее в реализации
2. хорошо параллелизуется адаптируется к неоднородному освещению сцены

недостатки:

1. не может выполняться «на месте»
2. плохо работает на объектах большой (>N) толщины
3. В чем разница между параметром "порог" прямого преобразования и параметром "константа" адаптивного?

В отличие от обычного порогового преобразования адаптивный метод сравнивает яркость пикселя не с фиксированным порогом, а с некоторой средней яркостью в окрестностях этого пикселя. Адаптивный способ плохо срабатывает на однородно закрашенных областях, то есть малые шумы в изображении будут умножаться. С параметром "константа" пиксель должен быть ярче, чем среднее минус “C” (константа).

В результате, используя способ с константой “C” текст будет выглядеть гораздо четче, чем при параметре "порог" прямого преобразования.

Отличие между параметром "порог" прямого преобразования и параметром "константа" адаптивного в том, что при прямом преобразовании задаём пороговое значение, которое определяет минимум (пиксели преобразуются в чёрный). При адаптивном преобразовании будет рассчитываться значение пикселя в пределах его "фрейма" (появляется адаптивность и изменчивости).

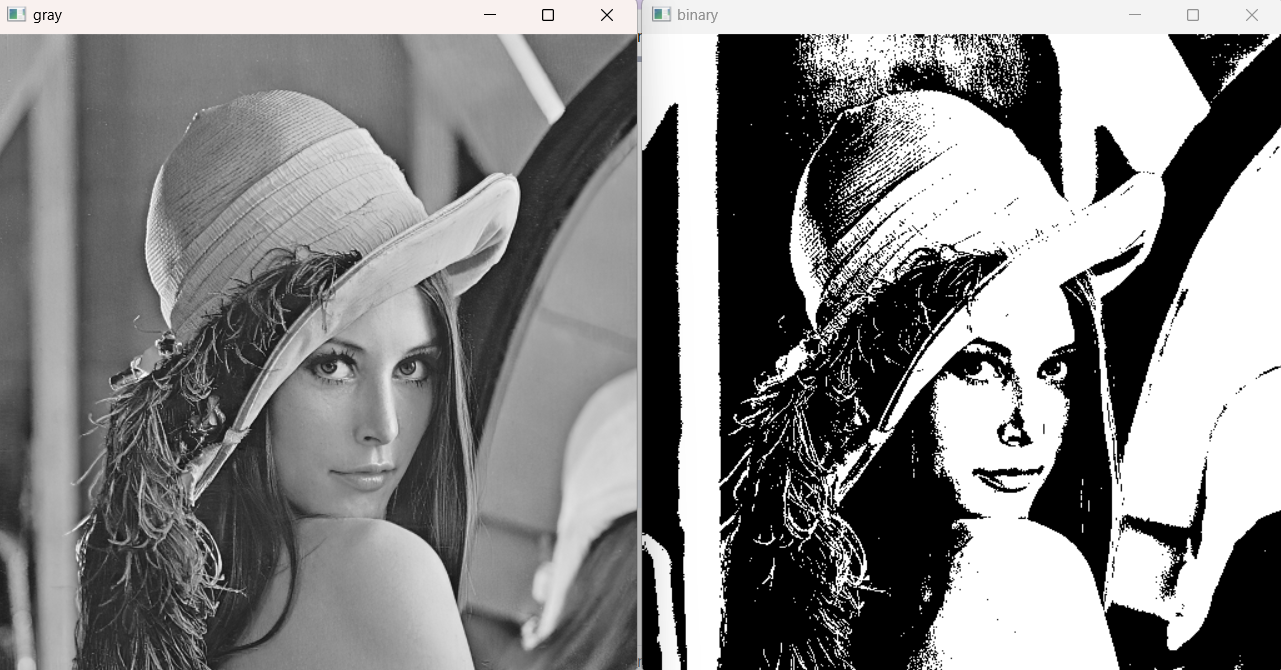
Вывод в консоль.

##### **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.

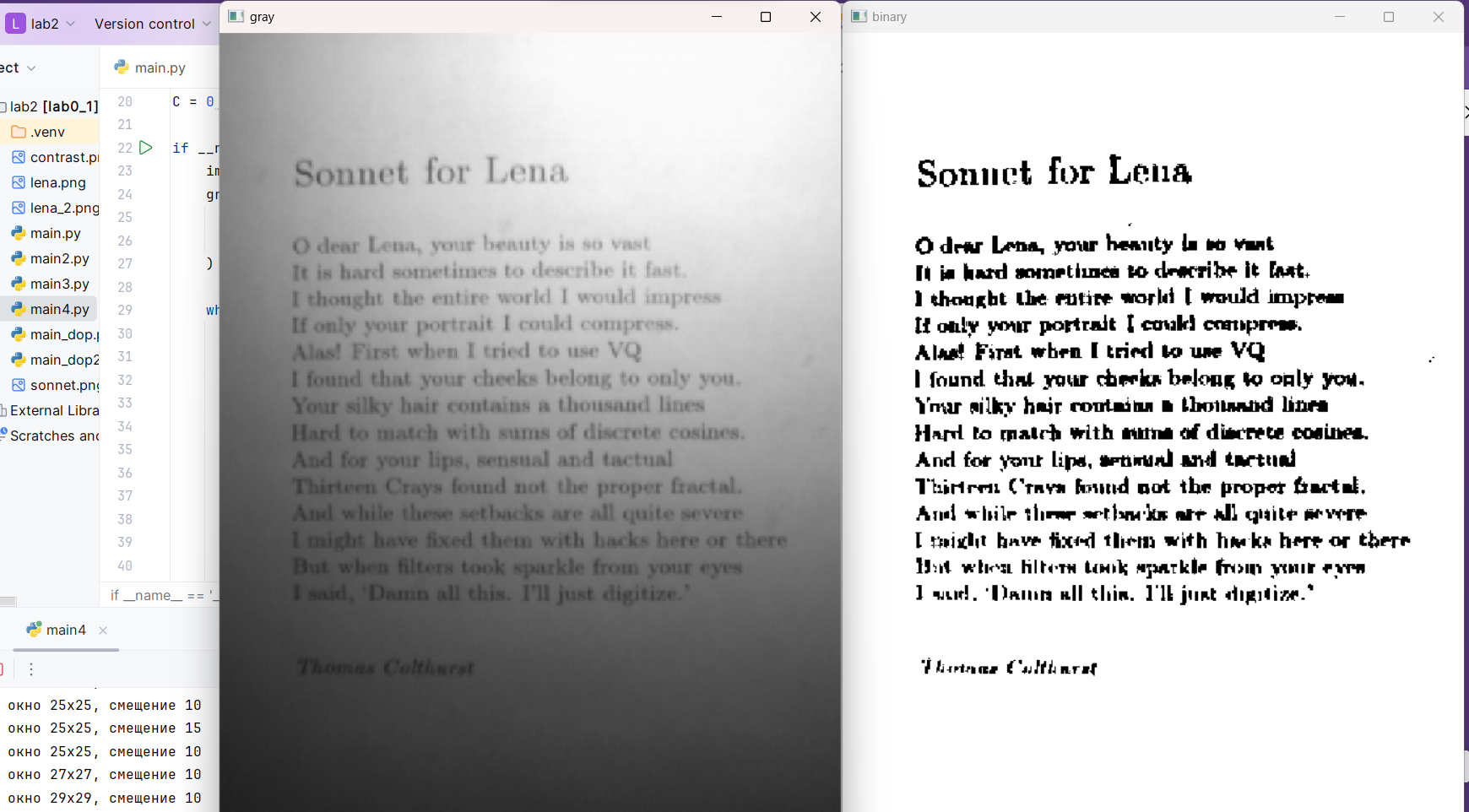


##### **Задание 4**

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.

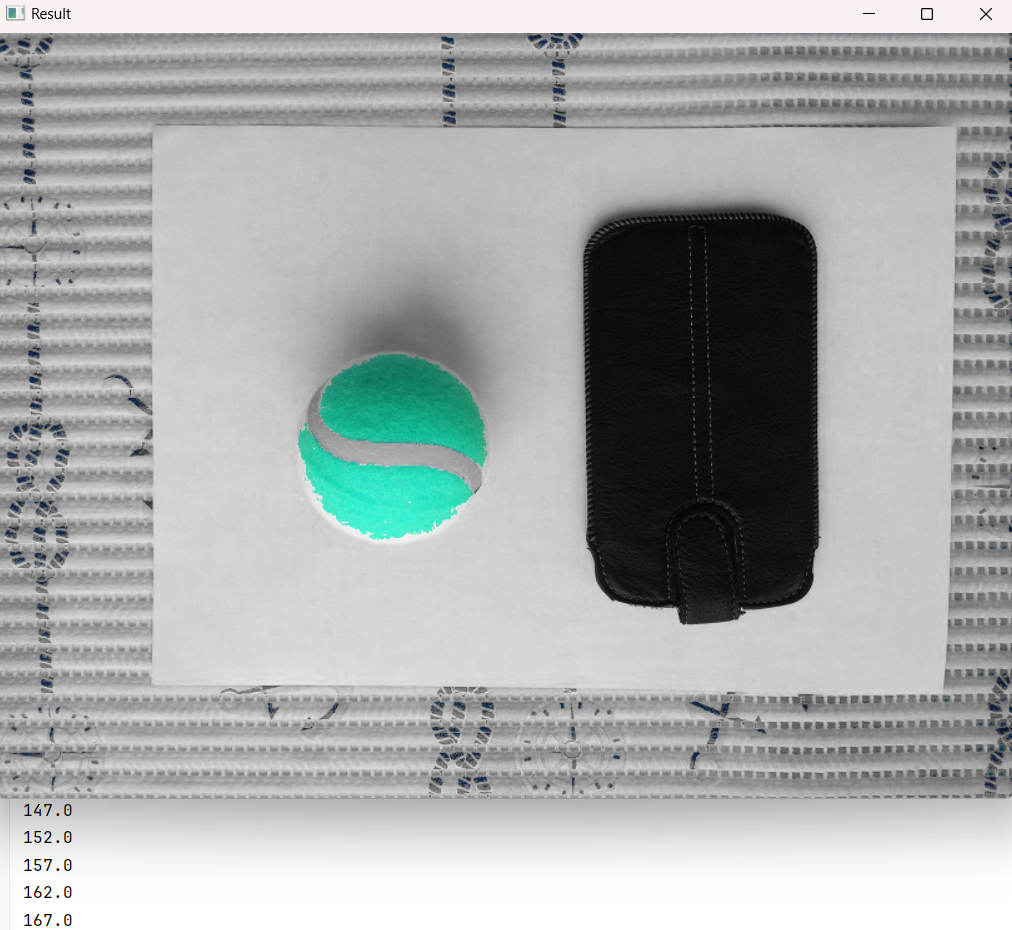
После каждого изменения параметров программа должна вывести их текущие значения.



##### **Дополнительное задание 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 задание 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.

Выводите полученные изображения на экран вместо бинарного изображения.

