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 №4.

# Проективная геометрия.

Выполнили студенты Копосов Лев Владимирович

Копосов Владимир Владимирович

Группа 22-ИСбо-1б

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

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

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

Кострома

2024

Вопросы:

1. Какие виды преобразований входят в следующие категории: движения, подобия, аффинные?

В категории: движения, подобия, аффинные входят следующие виды преобразований:

движения: параллельный перенос, поворот;

подобия: масштабирование;

аффинные: сжатие, параллельный перенос.

2. Чем отличаются функции cv2.warpPerspective() и cv2.perspectiveTransform()?

Функции cv2.warpPerspective() и cv2.perspectiveTransform() отличаются тем, что cv2.warpPerspective() применяет перспективное преобразование к изображениям, а `cv2.perspectiveTransform() преобразует координаты точек с использованием матрицы преобразования перспективы для работы с точками на изображениях из одной системы координат в другую.

cv2.warpPerspective() - создаёт новое изображение, в котором исходное изображение преобразуется с учетом выбранного преобразования. (изменяет его форму, размер и ориентацию).

cv2.perspectiveTransform() используется для преобразования координат точек с использованием матрицы преобразования из одной системы координат в другую.

3. Сколько точек позволяют однозначно описать аффинное преобразование? Проективное преобразование (гомографию)?

Для описания аффинного преобразования необходимо 3 точки.

Для описания проективного преобразования необходимо 4 точки.

4. Какую проблему при поиске преобразования позволяет решить метод RANSAC?

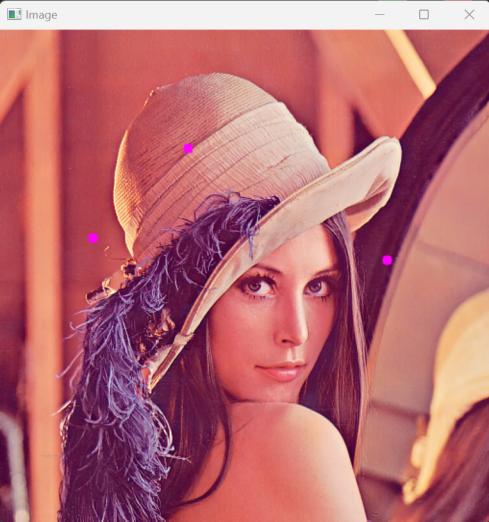
При поиске преобразования метод RANSAC предотвращает наличие выпадающих точек (выбросов).

Вывод в консоль.

Задание 1

Создайте приложение на Python, которое выполняет следующие операции: Загружает изображение из заданного файла Создаёт именованное окно с помощью функции cv2.namedWindow() и привязывает к нему реакцию на события мыши с помощью cv2.setMouseCallback(). Отображает загруженное изображение в окне с помощью cv2.imshow(). Помещает координаты щелчков левой кнопкой в список Щелчок правой кнопкой должен удалить последнюю запомненную позицию, если она есть. На месте записанных щелчков должны отображаться маркеры выбранного вам вида (круги, крестики и т.п.) По нажатию Esc окно должно быть закрыто вызовом cv2.destroyNamedWindow(). Список должен быть выведен на экран.

Рекомендации: оформите данные об окне, обработчик событий и хранилище данных в виде одного класса. Для упрощения работы с классом можно использовать механизм "менеджер контекста".



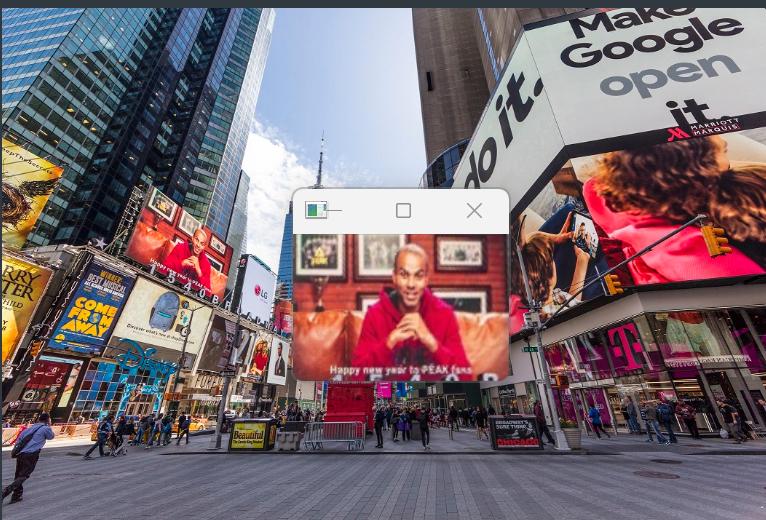
Задание 2

Загрузите изображение-панораму times-square.jpg, и, используя класс из задания 1, получите координаты четырёх точек на этой панораме. Эти точки должны являться вершинами четырёхугольника, начиная с левой верхней, по часовой стрелке.

Затем используйте функцию cv2.getPerspectiveTransform(), чтобы получить перспективное преобразование, позволяющее превратить выбранный четырёхугольник в прямоугольник.

Размер прямоугольника определяйте как максимум из "горизонтальных" и "вертикальных" сторон выбранного четырёхугольника.

С помощью функции cv2.warpPerspective() подвергните панораму полученному преобразованию, и выведите результат на экран. У вас должен получиться крупный план выбранного фрагмента панорамы с поправкой на перспективу.



Задание 3

Загрузите произвольное изображение. Затем загрузите панораму times-square.jpg, и, используя класс из задания 1, получите координаты четырёх точек, как в задании 2. Используйте функцию cv2.getPerspectiveTransform(), чтобы получить перспективное преобразование, позволяющее вписать выбранное вами произвольное изображение в этот четырёхугольник. С помощью функции cv2.warpPerspective() подвергните второе изображение полученному преобразованию, и наложите результат на первое изображение.

Для этого вам потребуется подготовить маску - массив логических значений, равный по размеру панораме и имеющий 3 канала. С помощью функции cv2.fillPoly() заполните в этом массиве четырёхугольник, выбранный пользователем вначале. Перед заполнением смените тип данных массива на numpy.uint8, а после - на bool с помощью присваивания свойству .dtype. Обратите внимание, что массив вершин N-угольника для cv2.fillPoly() должен иметь специфический вид: форма 1xNx2, тип данных numpy.int32.

После этого используйте полученную маску для индексации нужных элементов изображений, как это было сделано в работе, посвящённой преобразованию Фурье.



Задание 4

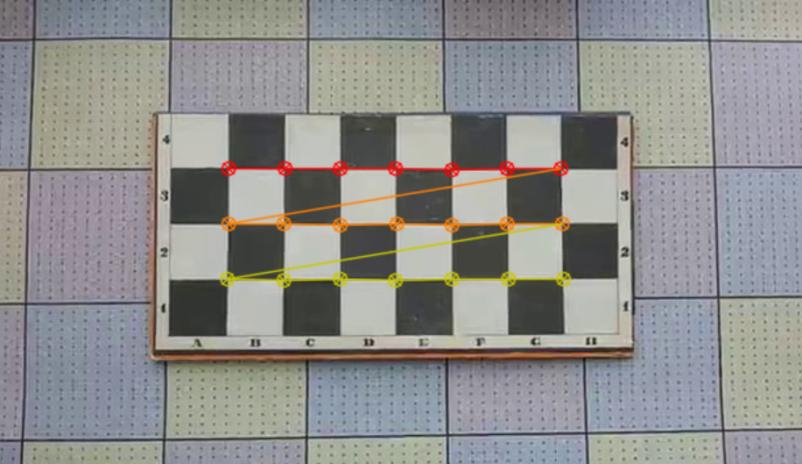
Используя класс cv2.VideoCapture, откройте приложенный видеофайл chessboard.mp4.

В цикле для каждого кадра, полученного с помощью метода .read(), используйте функцию cv2.findChessboardCorners() для нахождения шахматного поля. Размер шаблона на видео 7х3 (считаются не клетки, а внутренние углы). Укажите флаг cv2.CALIB\_CB\_FILTER\_QUADS.

Если нахождение было успешным, используйте функцию cv2.drawChessboardCorners() для отрисовки найденных углов на изображении.

Модифицированный кадр выведите в окне с помощью cv2.imshow(). Программа должна прерываться по завершению видео, или по нажатию пользователем клавиши Esc.

Обратите внимание на форму (.shape) массива углов - она представляет собой Nx1x2 где N - число найденных углов. Также обратите внимание, что иногда функция cv2.findChessboardCorners() может находить вершины не в порядке "сверху вниз и слева направо", а в обратном. Ваша программа должна обнаруживать такой случай и исправлять его, развернув массив углов в обратном порядке.



Задание 5

Модифицируйте программу из задания 4 следующим образом.

Загрузите произвольное изображение, и сгенерируйте шаблон шахматных углов для изображения такого размера (аналог углов, которые ищет функция cv2.findChessboardCorners()).

Для каждого кадра видео, найдите шахматный шаблон на изображении. Затем используйте функцию cv2.findHomography() в режиме RANSAC, чтобы найти преобразование, переводящее сгенерированный вами шахматный шаблон в найденный на видео.

Аналогично заданию 3, используйте функцию cv2.warpPerspective() с полученной матрицей для преобразования выбранного изображения. Затем произведите наложение преобразованного изображения на кадр и отобразите этот кадр в окне. Для генерации двоичной маски используйте функцию cv2.perspectiveTransform() на описании углов выбранного изображения, затем заполните маску с помощью cv2.fillPoly().

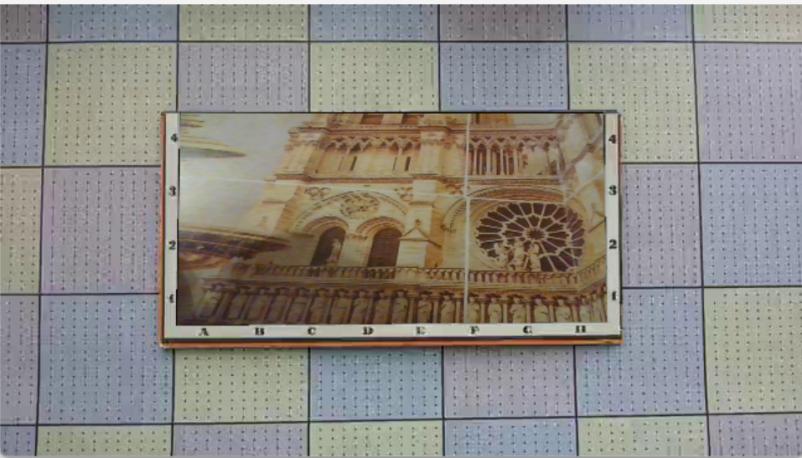
В результате на видео изображение шахматной доски должно быть заменено выбранным вами изображением.



Задание 6

Модифицируйте программу из задания 5 так, чтобы в качестве вставляемого изображения использовался кадр из другого видео. Чтобы использовать кадр с подключённой к компьютеру веб-камеры, передайте в качестве параметра конструктора cv2.VideoCapture целое число (например, 0). Альтернативно, можно указать имя файла или stream URL.

Чтобы узнать размер вставляемого кадра, используйте метод .get() класса VideoCapture со значениями параметра cv2.CAP\_PROP\_FRAME\_WIDTH и cv2.CAP\_PROP\_FRAME\_HEIGHT.



Дополнительное задание

Модифицируйте программу из задания 5 так, чтобы создать эффект "бесконечного туннеля", используя текущий кадр из видео в качестве вставляемого изображения. Количество итераций должно быть не менее трёх. Для получения полных баллов за дополнительное задание попробуйте обойтись одним вызовом cv2.findChessboardCorners() на кадр видео.