1. Скачайте пакет Python “Open\_VO” с репозитория на гитхабе используя “python/build/Open\_VO.cp312-win\_amd64.pyd” или “python/build/open\_vo-1.0.0-py3-none-any.whl”.
2. Создайте проект в PyCharm (или другой IDE) и создайте \*.py файл. Установите “Open\_VO”:
   1. “.pyd”: положите пакет “Open\_VO.cp312-win\_amd64.pyd” в ваш проект и импортируйте библиотеку “import Open\_VO”.
   2. “.whl”: установите пакет “pip install python/build/open\_vo-1.0.0-py3-none-any.whl” и импортируйте библиотеку “import Open\_VO”.

Для использования функциональности работы с преобразованиями, фильтрацией и коррекцией изображений, необходимо импортировать модуль «combined\_imgproc».

**Преобразование изображений**

Следующие функции разработаны для получения матриц трансформации между двумя изображениями, используя ключевые точки. Результирующая матрица может быть использована для объединения изображений в панораму, оценке позиции камеры, исправления перспективы изображения и др. Для получения матрицы, необходимо найти множества ключевых точек на паре изображений с использованием OpenCV:

orb = cv2.ORB\_create(nfeatures=500)  
# Find keypoints and descriptors  
kp1\_cv, des1\_cv = orb.detectAndCompute(image1\_bgr, None)  
kp2\_cv, des2\_cv = orb.detectAndCompute(image2\_bgr\_transformed, None)

Сравнение полученных множеств точек с использованием OpenCV:

bf = cv2.BFMatcher(cv2.NORM\_HAMMING, crossCheck=False) # Use NORM\_HAMMING for ORB  
knn\_matches\_cv = bf.knnMatch(des1\_cv, des2\_cv, k=2)

Преобразование множеств точек для дальнейшего их сопоставления:

src\_pts\_for\_transform = [ips.Point.from\_cv\_keypoint(kp1\_cv[m.queryIdx]) for m in good\_matches\_cv]  
dst\_pts\_for\_transform = [ips.Point.from\_cv\_keypoint(kp2\_cv[m.trainIdx]) for m in good\_matches\_cv]

**Вычисление матрицы гомографии**

Матрица гомографии является матрицей 3 на 3, описывающей проективную трансформацию между двумя плоскостями. Вычисление требует по меньшей мере 4-х пар ключевых точек, с помощью которых вы можете рассчитывать поворот, масштабные преобразования, сдвиг и дисторсию перспективы.

Для вызова функции, ей необходимо передать ключевые точки исходного и целевого изображений, а также, метод для фильтрации промежуточного результата:

H\_custom\_ransac = ips.find\_homography\_py\_custom\_ransac(  
 src\_pts\_for\_transform, dst\_pts\_for\_transform,  
 method\_filter="RANSAC)

Результатом является финальная матрица гомографии, которая может быть применена следующем образом на целевое изображение:

img1\_warped\_cv\_h = cv2.warpPerspective(image1\_bgr, H\_cv\_ransac,  
 (image1\_bgr.shape[1], image1\_bgr.shape[0]))  
cv2.imshow("Image1 Warped (CV RANSAC H)", img1\_warped\_cv\_h)

**Функции для вычисления матрицы афинных преобразований**

Матрица афинных преобразований между двумя изображениями является матрицей 2 на 3, описывающая афинные преобразования. Для её расчета необходимы по меньшей мере 3 пары ключевых точек, с помощью которых вы можете рассчитывать поворот, масштабные преобразования и сдвиг.

Для вызова функции, ей необходимо передать ключевые точки исходного и целевого изображений, а также, метод для фильтрации промежуточного результата:

A\_custom\_ransac = ips.estimate\_affine\_partial2d\_py\_custom\_ransac(  
 src\_pts\_for\_transform, dst\_pts\_for\_transform,  
 method\_filter="RANSAC)

Результатом является финальная матрица афинных преобразований, которая может быть применена следующим образом на целевое изображение:

img1\_warped\_custom\_a = cv2.warpAffine(image1\_bgr, A\_custom\_ransac,  
 (image1\_bgr.shape[1], image1\_bgr.shape[0]))  
cv2.imshow("Image1 Warped (Custom RANSAC A)", img1\_warped\_custom\_a)

**Коррекция цвета на изображении**

Следующей группой функций является группа, позволяющая корректировать цвет изображений.

**Функция корректировки гистограммы**

Данный метод осуществляет корректировку гистограммы изображения. Алгоритм основан на преобразовании функции распределения яркости (гистограммы) так, что результирующее кумулятивное распределение приближается к равномерному.

Для вызова функции, ей необходимо передать оригинальное изображение:

hist\_eq\_result\_custom = ips.apply\_histogram\_py(image1\_bgr.copy())

Результатом является скорректированное изображение:

cv2.imshow("Custom Global HistEq", hist\_eq\_result\_custom)

**Функция коррекции CLAHE**

Опция продвинутого выравнивания гистограммы.

Для вызова функции, ей необходимо передать оригинальное изображение, размер блока коррекции и относительное ограничение по высоте (максимальное значение) гистограммы на каждом блоке:

clahe\_result\_custom = ips.apply\_histogram\_clahe\_py(image1\_bgr.copy(), tile\_size\_x=80, tile\_size\_y=80, relative\_clip\_limit=3.0)

Результатом является скорректированное изображение:

cv2.imshow("Custom CLAHE", clahe\_result\_custom)