Homographies for Computer Vision

Introduction

Homographies play an important role in the field of computer vision. The points on a planar surface and the corresponding points on an acquired image of that surface are related by an homography. Also, the points on any two images of the same planar surface in space are related by another homography (assuming a pinhole camera model). This has many practical applications, such as distance measurements on a plane, image rectification, image registration/alignment, and several other. The pose of a camera relative to a planar surface can also be estimated from the homography that relates the points on that surface and the corresponding points in the acquired image.

In this lab work you will have the opportunity to use homographies for the following applications:

- Localize known objects (ex: partially occluded box – fig. 1.a)
- Measure distances on a plane with a single camera (ex: distances on a soccer field – fig. 1.b)
- Recognize objects using template matching (ex: augmented reality markers – fig. 1.c)
- Generate a bird's eye view of a lane (fig. 1.d)

a) b) c) d)

Figure 1 – Some applications of homographies: a) Localizing known objects; b) Measuring distances on a football field; c) Recognizing an augmented reality marker; d) Generating a bird's eye view of a lane.

General aims

To apply the knowledge about computer vision techniques, acquired in this course, namely, edge detection, segmentation, feature point detection and matching techniques, and geometric transformations, using the OpenCV library as development tool.

Tasks

1. Localize objects

- Develop an application that localizes an object in a scene, even when the object is partially occluded (fig. 2).
- Test different feature point detectors and different parameters for matching the points, namely, the value of the Lowe's ratio test and the influence of using the RANSAC method or not.
- Test with different objects and scenes; test also with non-planar objects.

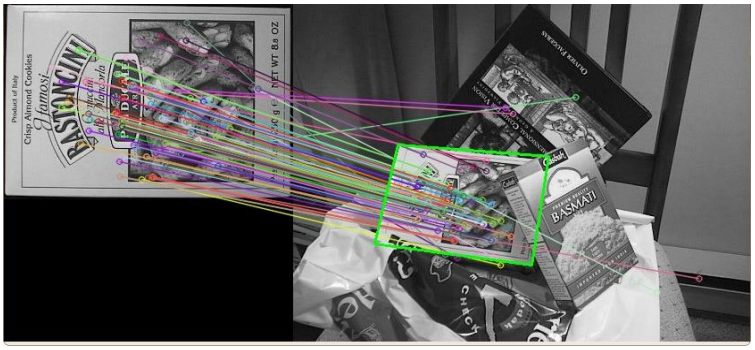

Figure 2 – Using feature matching and an homography to locate a planar object.

2. Measure positions and distances on a plane with a single camera

- Develop an application that allows the measurement of the positions of points and distances on a plane with a single camera.
- Take a photo of a plane, in perspective (/non-frontal) view, on which some points are marked whose coordinates are well known (figure 3).
- Using a subset of the points, calculate the homography that relates the points on the plane and on the acquired image.
- Calculate the position on the plane and the real distance (in mm) between some points on the plane.
- Notes:
  i. You may select manually the points in the image used to calculate the homography;
  ii. When calculating the homography, establish some wrong correspondence(s) and analyse the effect of using the RANSAC method or not.

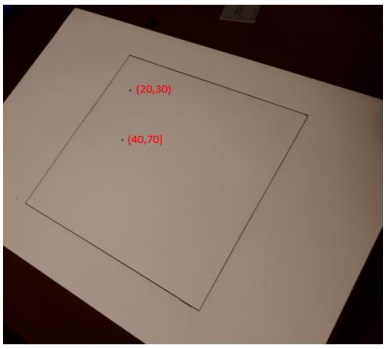

Figure 3 - Measuring positions on a plane with a single camera
(the text indicates the real world coordinates, in mm; origin is the top left corner of the square).

3. Recognize planar markers using template matching

- Develop an application that recognizes 2 reference markers, in an image taken from perspective view (ex: markers used in augmented reality – fig. 4); the application must draw, automatically, a red line around one of the markers and a green line around the other (fig. 4).
- Notes:
  i. You may use any markers, different from those used in the illustration; they may not be on the same 3D plane;
  ii. You have to detect the corners of each marker (in a first phase, you may pick the corners manually) and calculate the homography that maps each marker to a frontal view; after applying that homography, using the warpPerspective() function from OpenCV library, you may need to rotate the warped image to do the matching.

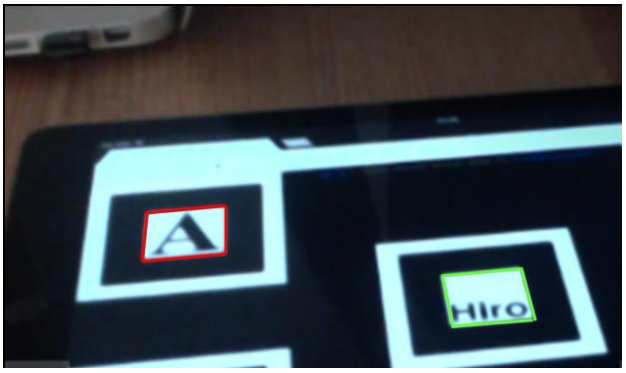

Figure 4 – Using template matching to recognize two different planar markers.

4. Generate a bird's eye view of a lane

- Develop an application that processes an image of a lane, as seen by a car driver (fig.5.a), generating a bird's eye view of the lane (fig. 5.b).

- Notes:
  i. You may use some simplification assumptions regarding the scene that have implications in the quality of the acquired images and in the image processing steps; indicate, in the report, the ones that you have used; you may acquire your own images.
  ii. For detecting the limits of the lane you may use both edge detection and segmentation techniques.
  iii. Use images in which the lower part of the image (representing the zone of the lane closest to the car) corresponds to a straight part of the lane.

a) b)

Figure 5 – a) Acquired image and detected straight lines (dotted yellow lines); b) bird's eye view of the lane.

Work development, report and submission

The work must be done by **groups of 3 students** (preferably of the same class/"turma").

A short report (max. 5 pages) must be submitted, in “pdf” format, including:

- any additional specifications (if needed);
- the description of the proposed global solutions, including illustrations of the results of the main intermediate steps;
- relevant comments about the efficacy of the used methods, describing the main problems that were encountered and any proposed solutions;
- the status of the proposed methods and the degree of fulfillment of the aims;
- an analysis of performance of the proposed methods, illustrated with some results.

Annexes may be included to show additional results that do not fit in the main report.

The report and the developed code (preferably in Jupyter notebook format), with meaningful comments and processed images, namely the images used in the illustrations of the report, must be submitted in a single 'zip' file, at the Computer Vision page in Moodle, until the end of **2023/Apr/06**.

Some useful links

- https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html
- https://docs.opencv.org/4.x/d7/dff/tutorial_feature_homography.html
- https://docs.opencv.org/4.x/d5/d6f/tutorial_feature_flann_matcher.html
- https://docs.opencv.org/4.x/dc/dc3/tutorial_py_matcher.html
- https://stackoverflow.com/questions/51197091/how-does-the-lowes-ratio-test-work
- https://www.geeksforgeeks.org/feature-detection-and-matching-with-opencv-python/
- https://learnopencv.com/homography-examples-using-opencv-python-c/
- https://pyimagesearch.com/2015/03/09/capturing-mouse-click-events-with-python-and-opencv/
- https://www.geeksforgeeks.org/opencv-python-tutorial/#feature
- https://www.geeksforgeeks.org/find-co-ordinates-of-contours-using-opencv-python/
- https://www.geeksforgeeks.org/image-registration-using-opencv-python/
- https://answers.opencv.org/question/18436/what-to-do-with-dmatch-value/
- https://blog.francium.tech/feature-detection-and-matching-with-opencv-5fd2394a590