ECE661: Homework 2

Fall 2022
Due Date: 1:30pm, Sep 08, 2022

In this homework, you will estimate homographies between a given set of images and use the estimated homographies to transform images. Note that you can NOT use any built-in functions to compute homography matrices or for image warping.

Turn in typed solutions via BrightSpace. Additional instructions can be found at BrightSpace.

1 Programming Tasks

You will be calculating a homography between two images by manually recording the pixel coordinates of a set of corresponding points in two images and using these coordinates to calculate the unknown elements of the homography that relates the images.

In what follows, Section 1.1 gives an overview of the problem definition. Section 1.2 has a list of recommended libraries and tools. The two programming tasks for this homework are defined in Sections 1.3 and 1.4. Section 1.5 has some additional tips to obtain more robust homography estimation. Finally, the homework submission instructions are given in Section 1.6.

1.1 Overview

The goal of this homework is to apply your conceptual understanding about homography estimation to transform one image onto another. For homography estimation, we need correspondence points between an image pair. For this homework, you will find the coordinates for the correspondence points using one or more of the recommended tools given in the following section. Then show your results for the tasks given in Sections 1.3 and 1.4.

1.2 Recommended Tools and Libraries

- **Programming languages:**

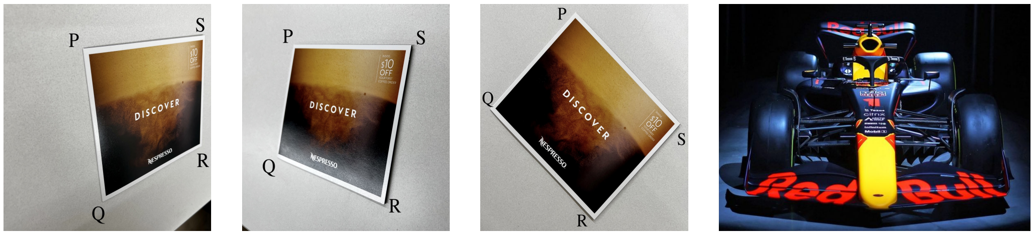
- Python with scikit-image or OpenCV. It's highly recommended to learn how to manage environments as opposed to installing libraries system-wide. Therefore, we want you to develop your code in an Anaconda [3] environment ("conda" for short) in order to get you ready for future homework assignments that would be easier to execute in such an environment.
- After you have implemented your solutions in your conda environment, we ask you to **submit the `environment.yml` file** along with your .py files. You can generate the .yml file as mentioned in the Tips below.

- **Tips on setting up your Anaconda environment:**

  A very useful cheatsheet on the conda commands can be found here [4]. Below is an abbreviated list of commands to get you started:

  - If you are used to using pip, execute the following to download Anaconda:
    `sudo pip install conda`
    For alternatives to pip, follow the instructions here [5] for installation.
  - Create your ECE661 conda enviroment:
    `conda create --name ece661 python=3.8`
  - Activate your new conda environment:
    `conda activate ece661`
  - Install the necessary packages (e.g. Scikit-Image, NumPy) for your solutions
    `conda install numpy scikit-image -c conda-forge`
  - After you have created the conda environment and installed the necessary packakges, use the following command to export a snapshot of the package dependencies in your current conda environment:
    `conda env export > environment.yml`

- **Tools to obtain a pixel coordinates in an image:**
  - GUI-based tools: GIMP [1] for Linux users and IrfanView [2] or GIMP for Windows users.
  - Command-line tools: ImageMagick [6] has a nice collection of command-line image manipulation tools. Alternatively, if you don't want to install anything on your system, simply load an image with python and plot using Matplotlib, when you hover your mouse pointer over the plotted image you can see the point coordinates in the status bar.



(a) View1 of the card image
(b) View2 of the card image
(c) View3 of the card image
(d) Car image for mapping

Figure 1: Images for Task1. Note that both outer or inner corner points of the given card images are acceptable solutions. Download the input images provided for Task1 separately.

1.3 Task 1

You are given four images which are shown in Figs. 1a, 1b, 1c, and 1d. The first three images consist of a card on a table, while the fourth image depicts one of the fastest cars ever made. Complete the following tasks using these images.

1. Using just four points, pick a region of interest (ROI) from the car image in Fig. 1d and project that region on the PQRS frames shown in Figs. 1a, 1b, and 1c. For this task, you need to find the homographies between the following pairs of images: (1) images shown in Figs. 1d and 1a, (2) images shown in Figs. 1d and 1b, and (3) images shown in Figs. 1d and 1c.

2. Find homographies between images shown in Figs. 1a and 1b, and between images shown in Figs. 1b and 1c. Then apply the product of the two homographies to the image shown in Fig. 1a. The resulting image should look similar to the image shown in Fig. 1c.

3. Using only affine homographies, map the entire car image to the PQRS regions in all three card images. Report the results. If one of the results is better than the others, explain why.

You can use an image editor such as GIMP or IrfanView to determine the pixel coordinates of a point in an image.

1.4 Task 2

Repeat the steps of Task 1 using your own images. You can capture three images of a planar surface from three different viewpoints such as the ones shown in Figs. 1a, 1b, and 1c. For the fourth image you can obtain a picture of your choice (animal, celebrity, etc.) from the Internet or use a picture of your own.

1.5 Additional Notes

- To project your chosen ROI shown in Fig. 1d into the frame $PQRS$ you can draw a bounding box $P'Q'R'S'$ around in Fig. 1d and estimate the homography using the corresponding pairs of points. In this case PP', QQ', RR', and SS' are the corresponding pairs of points. However, using just four correspondences for computing a homography is likely to give you very poor results. When using a system of linear equations for calculating the unknowns, a general rule of thumb is that you need five times as many equations as the number of unknowns. For this homework, we just want you to experiment with increasing the number of equations though additional correspondences between the two images of an image pair. For example, you can double the number of equations by using the endpoints of the perpendicular bisectors of the bounding boxes as shown below (see Fig. 2). Use such ploys to whatever extent you wish and see how that affects the quality of your homographies.
- For debugging your code efficiently, plot intermediate output at every important step and verify it visually. For example, after recording pixel coordinates manually, plot those points on your input images and verify visually if those points are correct.

1.6 Submission Instructions

Include a typed report explaining how did you solve the given programming tasks.

1. Turn in a zipped file with a typed pdf report with source code files. Include your input and output images in the report itself. Rename your .zip file as hw2_<First Name><Last Name>.zip and follow the same file naming convention for your pdf report too.

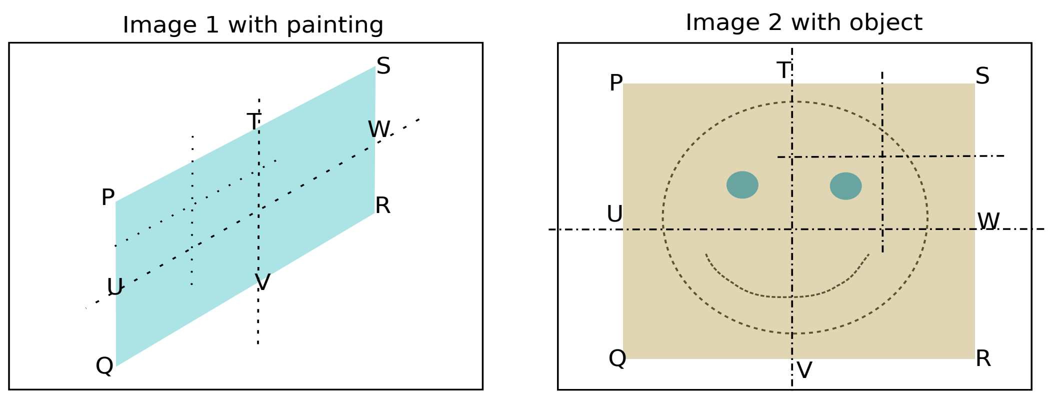


Figure 2: Increasing the number of correspondences between a pair of images. Note that the smaller bisectors are just symbolic. They show how to obtain larger number of correspondences with just four manually picked points.

2. Your pdf must include a description of
   - The logic that you used to solve the given tasks.
   - The steps that you used to compute the homographies including equations.
   - Your own input images for Task 2 and output images for the two tasks.
   - Your source code. Make sure that your source code files are adequately commented and cleaned up.
3. Indicate the points that you used on each image to obtain the homographies.
4. In order to avoid large file size of your submission, include JPEG images in your report for showing your results and your input images for Task2.
5. The sample solutions from previous years are for reference only, it's important not to get too biased by those solutions. You're free to format your report however you like as long as it addresses all the required components in the given programming tasks. There are many possible ways to format your report. For example, you're free to generate your pdf report using Jupyter notebook with inline explanation and results, and include your python code (.py files) separately. **Your code and final report must be your own work.**

References

[1] GNU Image Manipulation Program. URL https://docs.gimp.org/2.10/en/.

[2] IrfanView. URL https://www.irfanview.com/.

[3] Anaconda – Managing Environments, . URL https://docs.conda.io/projects/conda/en/latest/user-guide/getting-started.html.

[4] Conda Cheat Sheet, . URL https://docs.conda.io/projects/conda/en/4.6.0/_downloads/52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf.

[5] Conda Installation, . URL https://conda.io/projects/conda/en/latest/user-guide/install/index.html.

[6] ImageMagick – Command-line Tools. URL https://imagemagick.org/script/command-line-tools.php.