ECE 5460:
Image Processing
Fall 2023
Ertin

Project 3

Posted: October Tuesday 31, 2023
Due: Friday 23:59, November 10, 2023

We recommend you use the basic Python notebook discussed in the tutorials. You can run it locally or on Google Colab. To use Colab, upload it to Google Drive and double-click the notebook (or right-click and select Open with Google Colaboratory), which will allow you to complete the problems without setting up your own environment. Once you have finished, make sure all the cells are run before downloading the notebook).

Submission Instructions: Please submit three categories of files on Canvas: (1) the Python jupyter notebook with the relevant source code and with all the cells run, and (2) The input output images in jpeg format (3) A typed !! (latex, word, etc) report saved as as a pdf file that includes all input and output images and a brief description the processing details and description of each input/output image.

Late Submission Policy: As discussed in the late policy. Due dates will be strictly enforced. 20% penalty for every late day. Solutions will be posted to the course web page in 5 days. After solutions are posted, no credit will be issued for late work.

1 Introduction

This is the first part of a two part project. The goal of this first part is to introduce you the use of homographies with an example on image mosaicing and image warping. You will take two or more photographs and create an image mosaic by registering, establishing homographies, projective warping, resampling, and blending your results.
The steps of the assignment are:

- Capture images (2 pts)
- Recover homographies (2 pts)
- Warp the images (2 pts) [produce at least two examples of rectified images]
- Blend images into a mosaic (20 pts) [show source images and results for two mosaics.]

1.1 Capture Images

Shoot photographs so that the transforms between them can be expressed as homographies. One way to do this is to shoot from the same point of view but with different view directions, and with overlapping fields of view. Another way to do this is to shoot pictures of a planar surface (e.g. a painting or a bookcover) or a very far away scene (i.e. plane at infinity) from different points of view.
The easiest way to acquire pictures is using a digital camera. Make sure to use the highest resolution setting. Read the image into your python notbeook. We expect you to acquire at least two images yourself, but you are free to supplement the photos you take with other sources (e.g. old photographs, scanned images, the Internet).
Here are some recommendations from Prof. Alexei Efros on taking good photos for processing.

- Avoid fisheye lenses or lenses with significant barrel distortion (do straight lines come out straight?). Any focal length is ok in principle, but wide angle lenses often make more interesting mosaics.
- Shoot as close together in time as possible, so your subjects don't move on you, and lighting doesn't change too much (unless you want this effect for artistic reasons).
- Use identical aperture & exposure settings, if possible. It's nice to use identical exposures so that the images will have identical brightness in the overlap region. You can use your smartphone camera, but please make sure the camera setting does not change. To do this, you can use exposure and focus locking (AE/AF) (e.g. in iPhone, you can press and hold to do this). You can also use a camera app. Please check the EXIF data of the photos once they are taken. Overlap the fields of view significantly. 40% to 70% overlap is recommended.

1.2 Estimate the Homography Matrix

Before you can warp your images into alignment, you need to recover the parameters of the transformation between each pair of images. In the special case we consider , the transformation is a homography as discussed in class, represented by a matrix 3×3 H with 8 degrees of freedom. Write a function that recovers the homography matrix from matching pair of points `image1points`, `image2points` using the total least square approach discussed in the class. Establishing point correspondences automatically will be covered later. Here we follow the manual way of providing point matches is with a mouse-clicking interface. You can write your own using the bare-bones mouse input function. Click x points on first image and click corresponding x points on the second image in the same order. (`matplotlib.pyplot.ginput`)

1.3 Image Warping and Rectification

Now that you know the parameters of the homography H, you need to warp your images using this homography. Write a function that takes a given image and warps it using H. You can use either forward of inverse warping. You can use built in interpolation functions. Pay attention to is the size of the resulting image (you can predict the bounding box by transforming the four corners of the image through H and making sure that the output image can accommodate those coordinates). Think how you can mark pixels which don't have any values. Consider using an alpha mask (or alpha channel) to represent empty and occupied pixels.

Take a few sample images with some planar surfaces, and warp them so that the plane is frontal-parallel (e.g. like the tile floor examples from the paintings we discussed in class). Note that since here you only have one image and need to compute a homography for, say, ground plane rectification (rotating the camera to point downward), you will need to define the correspondences using something you know about the image. For example, suppose your image contains tiles on the floor that are square, you can click on the four corners of a tile in the image and store them in `im1_pts` while `im2_pts` you define arbitrarily to a square pixel grid of your choice.

1.4 Forming Mosaics

Now take two (or more) images you took with overlap, and create an image mosaic. Instead of having one picture standing in front of the other, you have to use blending (weighted averaging). You can leave one image unwarped and warp the other image(s) into its projection, or you can warp all images into a new projection.

If you used an alpha channel, you can apply simple feathering (weighted averaging) at every pixel. Setting alpha for each image takes some thought. Research on methods for blending with varying weights (for example). For extra credit you can use sophisticated blending techniques, such as a Laplacian pyramid.