Project 1 : Homography

In this project you will learn about homography matrices and use them to warp planar regions in images. The project consists of two major parts: rectification of a single planar region, and compositing of one planar region onto another.

All image parameters for skeleton functions are passed as 3D numpy arrays with shape (height, width, 3) for color images and (height, width, 1) for greyscale.

You will need to compute the eigendecomposition of a real-valued symmetric matrix. Numpy provides such a function: Numpy Eigendecomposition. **NOTE** that this function returns the eigenvectors as *columns*, not rows as one might expect. The i'th eigenvector is evecs[:, i].

Skeleton

Your grade will be based on code that you add to complete the skeleton file linked below. Though it should not be necessary, you are free to additional functions as you see fit to complete the assignment. You are not, however, allowed to import additional libraries or submit separate code files. Everything you will need has been included in the skeleton.

All image parameters for skeleton functions are passed as 3D numpy arrays with shape (height, width, 3) for color images. **You should return images in the same format.**

The included `rectify.py` and `composite.py` files can be used to run the skeleton on the test images. Usage is as follows:

```
===[ Rectification ]===
USAGE: python rectify.py IMAGE TARGET_POINTS_FILE

OR
    python rectify.py IMAGE SOURCE_POINTS_FILE TARGET_POINTS_FILE

(You provide text files with points, one per line, in the form
```

```
    X1 Y1
    X2 Y2
    X3 Y3
    X4 Y4

     If SOURCE_POINTS_FILE is not given, and OpenCV is available, you will be
     prompted to click 4 points in a planar region.

===[ Compositing ]===
USAGE:
        python composite.py SOURCE TARGET MASK
OR
        python composite.py SOURCE TARGET SOURCE_POINTS TARGET_POINTS MASK

    SOURCE, TARGET, and MASK are paths to image files.

    MASK should be a greyscale image containing mask values for blending. White
    pixels in the mask correspond to SOURCE pixels, black pixels are TARGET
    pixels, and grey values are blends of the two.

    If SOURCE_POINTS and TARGET_POINTS are given, they should be paths to text files containing
    corresponding coplanar points in each image to be composited. The file should contain one
    point per line with a space separating the x and y coordinates:
        X1 Y1
        X2 Y2
         ..
        Xn Yn

     If the point files are not given, and OpenCV is available, you will be prompted
     to click corresponding points in planar regions in the two images.
```

Running the following command:

```
python rectify.py laptop.png laptop_screen.txt laptop_rect.txt
```

will compute the test rectification image. Running the following command:

```
python composite.py panda.png laptop.png panda_points.txt laptop_screen.txt panda_mask.png
```

will compute the test compositing image.

Output images are saved as "rectified.png" and "composited.png".

When you run the code with your own images, you will not be able to use the provided correspondence files. Instead, you will need to provide your own file with X/Y coordinates of corresponding points in each image (or use the built-in coordinate picker). The file format is one X/Y coordinate pair per line, corresponding pixels (for compositing) should be given in the same order in both files. If you don't have OpenCV available, most image editors will show you X/Y coordinates somewhere when you move your mouse cursor over the image.

Downloads

- Rectification: rectify.py
- Compositing: composite.py
- Skeleton: p1_skeleton.py
- Point Picker Tool: util.py
- Images: laptop.png panda.png panda_mask.png
- Correspondence Files: laptop_screen.txt laptop_rect.txt panda_points.txt

What to Submit (Drexel Learn)

Zip or tar.gz containing the following:

- **Code**: Include your completed p1_skeleton.py file.

- **Data**: Include any additional images you used for the assignment in the submission. If you use the provided point picker interface, you need to include the correspondences you selected as well. They are printed out after you have picked them.
- **Report (PDF)**: Writeup about your experience (what was difficult, etc.). Describe your experiments if you did any (e.g., changing parameters and showing their effects on the results; good scientifically conducted experiments will earn extra credits), and tell us what you did for extra credit. Include lots of pictures.

  When you run the code with your own images, you will not be able to use the provided correspondence files. Instead, you will need to provide your own file with X/Y coordinates of corresponding points in each image. The file format is two pairs of X/Y coordinates per line, one for the start image and one for the end image, separated by spaces. Submit all point list files used to generate images in your report.

Grading

- **Code** [60pts]
  - [20 pts]: Homography Functions. Fill in the build_A and compute_H functions in the skeleton.
  - [20 pts]: Warp Function. Complete the warp_homography function to warp an image with a homography.
    - [10 pts]: Proper application of the homography and pixel warping.
    - [10 pts]: Implement and use bilinear interpolation for pixel color lookup.
  - [10 pts]: Blending. Implement linear blending between two images using a mask (blend_with_mask).
  - [10 pts]: Rectify and Composite Functions. Call the correct functions implemented above to rectify and composite images.
- **Report** [10pts]
  - [10 pts]: Include the report PDF described above. Demonstrate your code by rectifying at least one image of your choosing and compositing one image pair of your choosing.
- **Extra Credit** [50+pts]
  - [15 pts]: Implement Laplacian pyramid blending instead of linear blending.
    - You may import and use scipy's convolution and/or image blurring functions to compute the pyramids.
  - [20 pts]: Implement planar mosaicing.
  - [? pts]: Do something really interesting.

  You must show the results of your code for each extra credit portion.