CS 763/CS 764: Lab Two

Projective Transformations

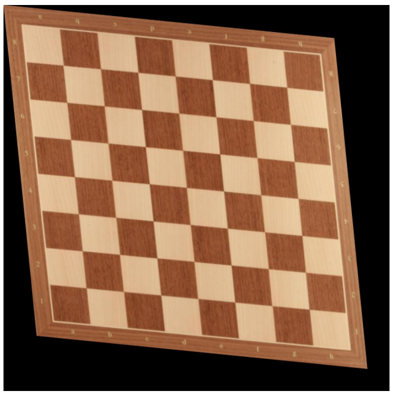
- Announced: 03 Feb. Due 11 Feb 9AM
- **This is a group assignment**

1 Geometric Transformations

In this lab we focus on some APIs in OpenCV to perform geometric transformations. Don't be misled by the 2D nature of this assignment, you should be able to connect it to the 3D problem.

1.1 Affine Transformations

Please perform the steps below in order.


(a) Distorted Chessboard

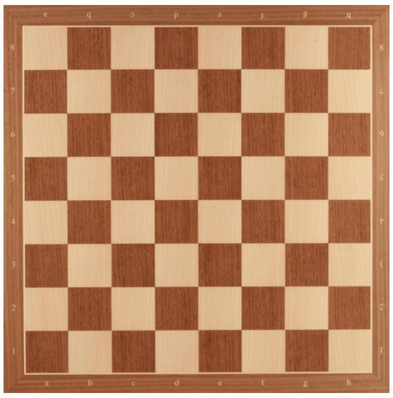

(b) Original Chessboard

Figure 1: Shear Transformation

1. Using OpenCV (`cv2.warpAffine()`) reconstruct the original image of chessboard of size 600x600 from the provided distorted chessboard image (`distorted.jpg`). It is to be assumed that this distortion is a planar axial shear distortion.

   In this part you need to construct the affine transformation matrix by yourself "manually" (e.g., using your scale) and feed it to the API. (Take a look into the documentation to get an idea about the format of the matrix).

   Q: You need to explain the method with which you obtained the original image (A trial-and-error brute force method is not expected.)

2. Now perform the "un-distortion" again, using the API **cv2.getAffineTransform()**. By "un-distortion" it is to be understood that the original image should be retrieved.

   Q: What were the differences, if any, you observed? How many point correspondences did you use?

Run Examples

- Manually computed affine tansformation:

  ```
  python3 affine-trans.py -mat manual
  ```

- API computed affine tansformation:

  ```
  python3 affine-trans.py -mat api
  ```

1.2 Perspective Transformations

A mysterious obelisk had appeared in the desert for a brief period, only to sink again in the sands soon after. This obelisk had a map that could lead to a great treasure. Fortunately, an on-sight archaeologist managed to capture a snapshot before it sunk. However, your cartographer refuses to assess the map before it is presented to her in a compliant form: a 512x385, front facing orthographic projection.

- Using the captured photograph (obelisk.png) recover the map in its compliant form by performing a perspective transformation (using the API from `OpenCV`).

  Hint: Before you go to the cartographer asking which dimension is the length or which side is up, please remember - North side is up!

  Q: In what way does 3D geometry enter into the picture here? How many point correspondences did you use?

- **Animation:** You were able to generate a compliant map (*view 2*) by applying a perspective transform on obelisk.png (*view 1*). But your cartographer is not convinced that this is the same map as the one in the obelisk photo. She does not know what is meant by a perspective transformation or any transformation for that matter. So, she wants you to show to her the transformation process (*view 1* to *view 2*) as an animation.

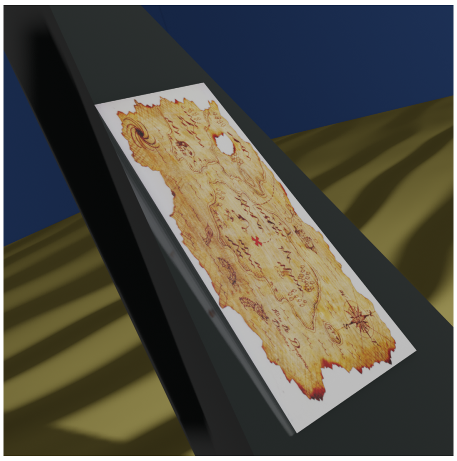
  To do so, you need to figure out intermediate views that can be shown in each frame. Make sure that the non-map part is always cropped/masked out (black) in each frame lest your cartographer focuses on the obelisk or the desert instead of the map.

  We expect to see a short animation (5 to 10 seconds) showing a slow transition from *view 1* to *view 2*. Only the map should be visible in the animation. Interpolation can be linear. Frame rate need not be exact, so there is no need to pre-compute the intermediate views before displaying the animation. But do supply us with a key to break the animation loop!

  Note: You do not have to store this as a movie anywhere. So, you should not need any library other than `OpenCV` and `numpy`.

  Q: Is the way to go from *view 1* to *view 2* unique? Please elaborate on your answer and don't forget to mention examples/counter-examples supporting your claim.

- Run format: `python3 persp-trans.py`


(a) Mysterious Obelisk

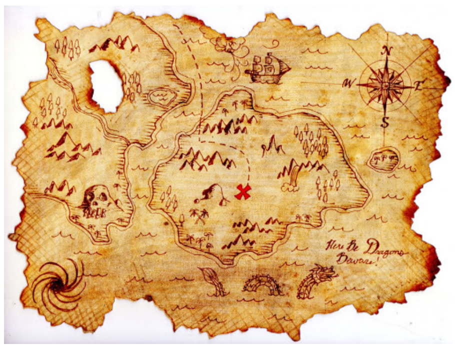

(b) Expected View of Map

Figure 2: Perspective Transformation

2 Document Scanner

Here you are going to implement a document scanner (like the one at `camscanner.com`) to (ideally) automatically detect correspondences for a document and transform it into a more readable format.

- First perform this task by manually identifying the correspondences. We are expecting this as a minimum.
- Now automate the selection task by using contour detection and the following assumptions:
  - The document of interest is a convex quadrilateral
  - The document is the largest object in the image.
  - The user wants to see the document in portrait mode
- An example (scan.jpg) is provided but you should work with 3 or more different documents of your own in increasing order of complexity. Design your input so as to get your code "working" on either 1 input or 2 inputs, and "not working" on the others. By "not working", the understanding is that the "un-distortion" is not achieved. Remember, the operation is supposed to be automatic. We expect the results for this part to be saved in the results directory.
- Run: `python3 document-scanner.py -i ../data/scan.jpg`

Submission Guidelines

1. The top assignment directory should include the lab submission as detailed below.

   (a) Do include a `readme.txt` (telling me whatever you want to tell me including any external help that you may have taken). Don't forget to include your honor code here (or your name and roll number). All members of a group are expected to (electronically) sign the honor code and the percentages (see below). This is a text file, not `pdf`, not `docx`.

   The `readme.txt` will contain individual contributions of each of the team members. If the assignment is finally worth 80 marks as graded by the TA, then a contribution of the form `80, 100, 60` (in sorted order of roll numbers) will result in (respectively) marks `64, 80, 48`. Do this for each question separately. A person claiming 100% is basically saying that (s)he can reproduce the entire assignment without the help of the other team members.

   (b) `ReflectionEssay.pdf`: Should contain the explanation for all the questions implicitly and explicitly raised. Provide an output of a sample run. Explain what you learnt in this assignment, and how this assignment improved your understanding. What will someone who reads this gain? Can this be a blog post, which if read end-to-end someone not in your class (but in your batch) will understand?

   (c) A directory called `code` which contains all source files, and only source files (no output junk files). The mapping of code file to questions should be obvious and canonical.

   (d) A directory called `results` to store output that needs to be saved (this will typically be explicitly stated in the question).

   (e) A directory called `data` on similar lines to code, whenever relevant. Note that your code should read your data in a relative manner.

   (f) Source files, and only source files (no output junk files). Mac users please don't include junk files such as `.DS_store`. We are not using MacOS.

   (g) Create a directory called `convincingDirectory` which contains anything else you want to share to convince the grader that you have solved the problem. We don't promise to look at this (especially if the code passes the tests) but who knows? This is your chance.

2. Once you have completed all the questions and are ready to make a submission, prepend the roll numbers of all members in your group to the top assignment directory name and create a submission folder that looks like (for group but you get the idea) this
   `130010009_140076001_150050001_lab0X_description.tgz`

   Please stick to `.tar.gz`. Do not use `.zip`. Do not use `.rar`

3. Your lab submission folder should look something like:

```
130010009_140076001_150050001_lab02_projective/
├── ReflectionEssay.pdf
├── code
│   ├── affine-trans.py
│   ├── persp-trans.py
│   └── document-scanner.py
├── data
│   ├── distorted.jpg
│   ├── obelisk.png
│   ├── document-1.jpg
│   ├── document-2.jpg
│   ├── document-3.jpg
│   └── scan.jpg
├── results
│   ├── document-1-output.jpg
│   ├── document-2-output.jpg
│   ├── document-3-output.jpg
│   └── scan-output.jpg
├── convincingDirectory
└── readme.txt
```

4. Submission. Very very important.

   (a) The lexicographic smallest roll number (on google classroom) in the group should submit the entire payload (with all the technical stuff).

   (b) All other roll numbers submit only `readme.txt` as discussed above.