COMPUTER VISION ASSIGNMENT 1

JUSTIN DE WITT* (21663904)

18 August 2022

1 IMAGE ROTATION AND SCALING

In this section we would like to design a transformation which can be used to rotate, scale, or otherwise transform and input image. The transformation procedure involves mapping each coordinate of an input image, to an output image, using a specified matrix which defines the projection.

1.1 Projective Transformations

In its most simple form, a projective transformation is defined as follows:

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

where the matrix composed of $[h_{11}, \ldots, h_{33}]$ defines the projection.

1.2 Inverse Mapping (Optional Read)

Similarly to the previous assignment, there are problems associated with forward projection when the magnitude of the determinant associated with the projection is larger than 1. This implies that the unit region in the input space, is smaller than a unit region in the output space. If we use forward projection in the context of image transformation, there will be a mesh-like grid of uninitialized pixels in the output region, resulting from images containing only discrete coordinates from which pixel data can be obtained. As a simple example, consider the matrix :

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

and the input vectors [1, 0, 0] and [2, 0, 0], corresponding to successive pixels in the input image. The mapping is [2,0,0] and [4, 0, 0] respectively, which neglects the odd numbered points. To solve this problem, we use an inverse mapping system with bilinear interpolation, discussed in the previous assignment.

1.3 Process

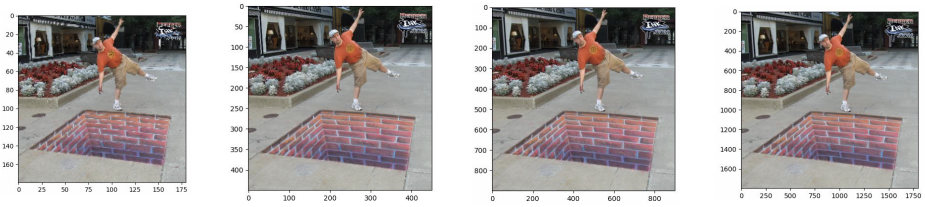
Once a transformation matrix is specified, and an input image provided, the 4 corners of the input image are forward-transformed to their corresponding location in output space. Thereafter, we computed the box which most tightly bounds these 4 coordinates, and use this box to initialize a blank image, which we will fill with data from the original image, using the inverse matrix. **Note**: any offset specified in matrix will not effect the final image result, the reason being that a bounding

box is *always* used to contain the output image. The bounding box encapsulates the image irrespective of the images potential offset. Essentially, we "cut off" the offset which was added during the mapping. Once the bounding box is obtained, we determine the offset which we are cutting off (specified by the minimum x-value, and the minimum y-value of the images 4 corners). Once this mapping system is employed to connect coordinates in output space, to coordinates in the input space, we can simply inverse transform each pixel in output space, to the corresponding pixel in the original image, and use bilinear interpolation as required.

1.4 Results

1.4.1 Image Scaling

A scaling matrix is simply a symmetric matrix with the scaling constant on the main diagonal. Here we display the results for simple image scaling:

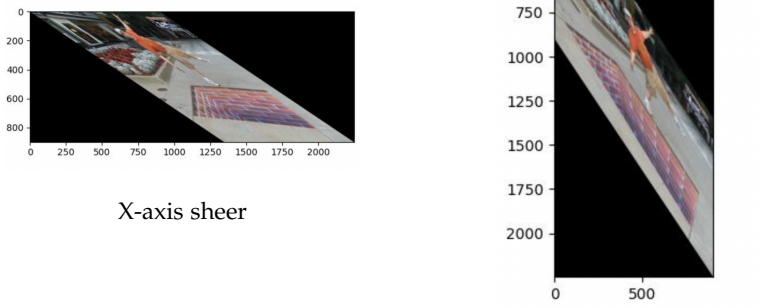


$\frac{1}{5}$x scale | $\frac{1}{2}$x scale | Original | 2x scale

Figure 1: Scale Transformations

1.4.2 Sheering Transformations

A sheering transform contains an positive value in the first row, second column, when sheering with respect to the x-axis. A sheer with respect to the y-axis contains a value in the opposing corresponding location. Here are sheering results:



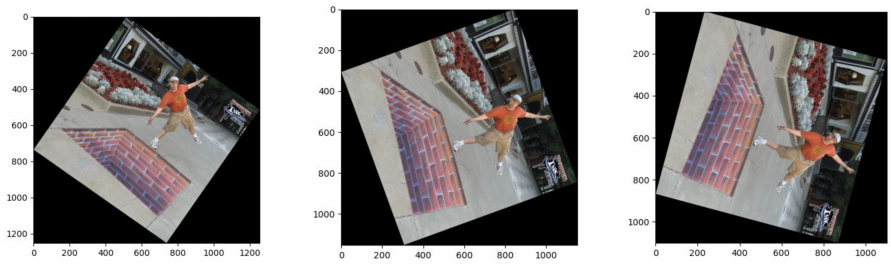
X-axis sheer

Y-axis sheer

Figure 2: Sheer Transformations

1.4.3 Rotation Transformations

We can also rotate an image either clockwise or counterclockwise by specifying the corresponding matrix. A rotation matrix can either be manually derived or found online, the results are shown below:



35deg rotation 70deg rotation 105deg rotation

Figure 3: Counterclockwise Rotation

1.4.4 General Transformations

It is a property of matrix transformations that applying transformations in succession (e.g. scale then rotate), is not different than simply applying the product of these matrices. This result can be used to combine the effects of different transformations.

2 IMPOSING PERSPECTIVE DISTORTION

In this section we would like to alter the perspective of a provided image, so that it can be suitably placed on a target image and match the context of placement. As an exercise, we are provided an image of a building, and we would like to plaster a movie poster on the side of the building, for marketing purposes.

2.1 Changing the Perspective of the Poster

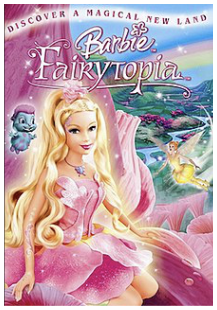
If we knew the matrix associated with the transformation that correctly scales and orients the poster to fit the building, we could use the methods of section 1 to transform the image accordingly. We should subsequently search for methods to obtain the desired transformation. Luckily, there is already a method to derive the matrix, H, using point correspondences. To begin the derivation, we locate 4 points on both the input and output images, that should correlate to one another once the transformation is complete. In this context, the 4 chosen points are the borders of the input image (movie poster) and the desired destination for these coordinates (the borders of the building face). For each point correspondency, we begin by trying to solve this system of equations:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ \dots \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

which relates a point (x_1, y_1) in the input image, to a corresponding point (x_1', y_1') in the output image. This system generalizes downards, where we can add more rows to the matrix A, and subsequently increase the dimension of the zero vector, to relate more correspondences. Solving for this system by hand is very difficult, but fortunately we can notice that this matrix is not full rank, the column-space and row-space are not the dimension. This implies, using the rank-nullity theorem, that the null-space is not of dimension 0. We can solve this system (for $\mathbf{H}$) be getting the nullspace, which is accomplished using the linear algebra library in *NumPy*.

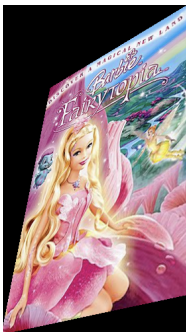
2.2 Transforming the Movie Poster

Once we have our 4 chosen point correspondencies, we can use the methods described above to transform our chosen movie poster. Consider the the images below:



Chosen Poster

Target Building



Transformed Poster

Result

Figure 4: Perspective Transformation Steps

Notice that the transformed poster contains black space. This comes as a result of the mapping, which inherently changes the perspective of the poster's view. As a result, we need to somehow bound the resulting poster by some box, so that it can be displayed in an image. The black region corresponds to excess space in the bounding box of the transformed poster, where image data is not present. To remove this region, we simply plaster the entire image, including the black trim, onto the target building, then simply replace all pure black pixels with the pixel found in the original building image at the same location. This will however remove black pixels from the poster as well. A simple work-around will be to choose a new padding colour, other than black.

2.2.1 *Aliasing*

If an image (movie poster) is significantly down-scaled during the perspective mapping procedure, there is a significant loss in image resolution/granularity as a result of the down-scale. Thin lines and details become almost dotted, and sparse, as a result of the resizing. The process happens because a high-volume space essentially gets forced into a low volume region, and to fill the low-volume region, values are sampled from the high-volume space. Two adjacent discrete coordinates in the low-volume region map to two (relatively) far apart coordinates in the original high volume space. It may be that these coordinates are so far apart, that there is detail between the coordinates that is overlooked. As a result, we have "aliasing" effects. The rendering of this movie poster was intentionally chosen as a similarly sized to the target building, which entirely combats aliasing effects.

3 REMOVING PERSPECTIVE DISTORTION

As shown in the previous section, we can distort an image to match the context of a target image. Is it possible to remove distortion effects, so that an image can be viewed from a different angel? That is what we experiment with in this section.

3.1 The Original Image

The image shown in section 1 is an example of "3D-street art". This art form is designed to give the illusion of caves, holes in the ground, or other 3D effects, when viewed from a certain position. The effects are not maintained when the beholder changes the perspective from which the artwork is observed. Consider the Original Image:

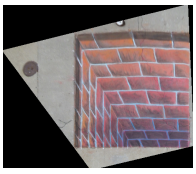
Figure 5: figure
Original Street Art Image

If we assume that the tiles on which the artist painted are squares, we can manually select the coordinates of the tile verticies, and map them to a perfect square, using methods from sections 1 and 2. The end result should be an image of the artwork, as if viewed from the top.

3.2 Results

We have already manually selected the coordinates corresponding to the verticies of the tile containing the artwork. From these four verticies, it is easy to determine the bounding box of the artwork. From this bounding box, we can also easily crop the image, to contain only the artwork, with little other noise in the image. We will then transform this cropped image using methods very similar to section 2. The mapping target is simply a 300 x 300 perfect square. The results are shown in the following figure.

Cropped Artwork



Vertical View

Cropped View

Figure 6: Artwork Perspective Transformation

4 IMAGE STITCHING

In this section we would like to combine two images taken of the same event, but from slightly different angles. The idea is to identify features which are present in both images, and use the techniques previously discussed to transform the positions of these features to correspond with the position of the features in the other image. Once the transformation is complete, the images can be stitched together, offering a wider view of the same event.

4.1 The Feature Correspondences Selected

The images are of a man kayaking down a river. The images are clearly taken at different moments in time due to the orientation of the man, and the man's position in the river. There are however features contained in both images. The chosen features are a pointy rock, a tall tree, a beaver dam in the background, and a small crevasse in a nearby rock. The features and images are shown in the following figure.

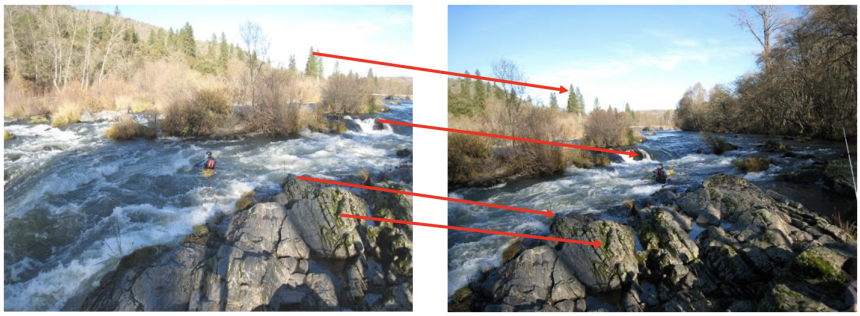


Figure 7: Feature Correspondences

For this procedure, we are planning to map the features from the left image, onto the features in the right image.

4.2 Applying The Transformation

Once-more, the steps used for this transformation are near identical to the previous three sections. After the 8 feature coordinates have been obtained, we can apply the same provided homography application function. The output from this function is shown:

Figure 8: Perspective Transformation of the Left Image.

4.3 Image Stitching

Once we have obtained the perspective adjusted image, we can attempt to combine these two images to form a single, larger image. There are two options for this:

1. Overlay the adjusted image onto the unadjusted image.
2. Overlay the unadjusted image onto the adjusted image.

For interest sake, both methods were implemented in this paper. The stitching was accomplished by determining the location of the feature closest to the origin (0,0) in the unadjusted image, and mapping it to the corresponding location in the adjusted image. We then "paste" the unadjusted image onto the adjusted image in this location. **Note**: before this procedure is implemented, the adjusted image is padded on the top, and on the right, to accommodate the addition of the unadjusted image. This process yields option 2) as specified above. If we want to revert to option 1), I found it easiest to simply iterate over the pixels of the unadjusted image again, and write the values over the adjusted image if the adjusted image contains a black pixel in this location. The two results are shown below.

Method (1) Method (2)

Figure 9: Image Stitching

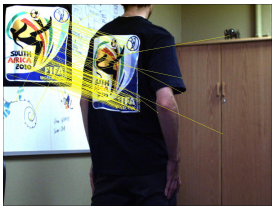
Notice that the river flows nicely throughout the stitched result. This shows that the procedure was somewhat successful in mapping the perspective from the leftmost image, to match that of the right image.

5 ESTIMATING HOMOGRAPHIES FROM SIFT FEATURE MATCHES

In this section we are provided a list of feature matches, obtained using the SIFT algorithm. Some of these "matches" do not actually correspond to matches, and should be removed prior to obtaining a homography matrix. To remove the noisy matches, we randomly sample 4 points to calculate a base homograph transform. We then apply this transform to each pair of SIFT coordinates corresponding to the template image. If the mapped location corresponds closely to the match identified by the corresponding SIFT point, the pair of points is added to a list. This process is repeated for 1000 randomly sampled sets of 4 initial points. This procedure is referred to as *random sample consensus (RANSAC)* algorithm. The longest output list is subsequently used to create a more accurate homography matrix, now composed of dozens of true matches, making it quite accurate. To illustrate this homography matrix, we plot the bounding box of the template image, and the location of the bounding box in the destination image, as multiplied by the obtained homography matrix.

5.1 Results

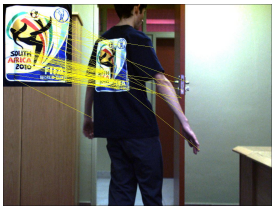
To tabulate the results, we first show the raw SIFT matches, then we plot the list of points obtained from the RANSAC algorithm plotted similarly. Finally, we show the bounding box of the template after transformation by the obtained homography matrix. For this implementation I have used a "match threshold" of 5 pixels, except for fifa11, which I got better results when chaning the threshold to 10.



Fifa1 Raw

Fifa1 RANSAC

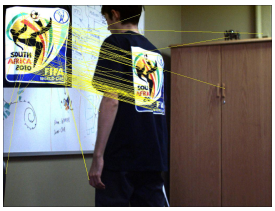
Fifa1 Bounds



Fifa2 Raw

Fifa2 RANSAC

Fifa2 Bounds



Fifa1 Raw

Fifa1 RANSAC

Fifa1 Bounds

Fifa4 Raw

Fifa4 RANSAC

Fifa4 Bounds

Fifa5 Raw

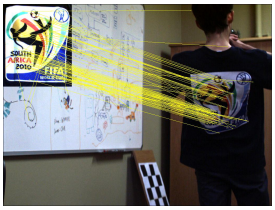
Fifa5 RANSAC

Fifa5 Bounds

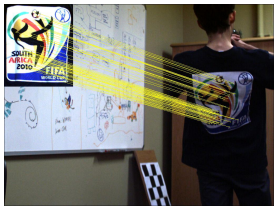
Fifa6 Raw

Fifa6 RANSAC

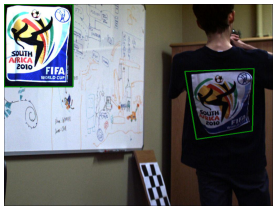
Fifa6 Bounds



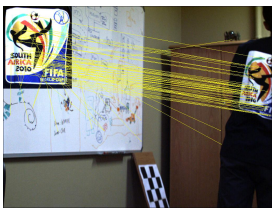
Fifa7 Raw



Fifa7 RANSAC



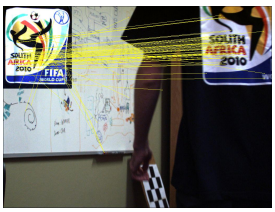
Fifa7 Bounds



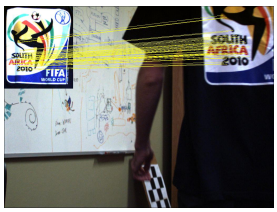
Fifa8 Raw

Fifa8 RANSAC

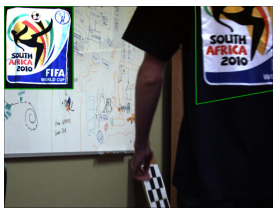
Fifa8 Bounds



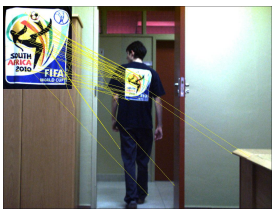
Fifa9 Raw



Fifa9 RANSAC



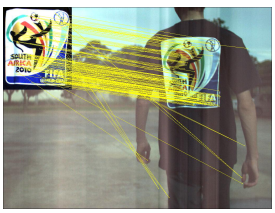
Fifa9 Bounds



Fifa10 Raw

Fifa10 RANSAC

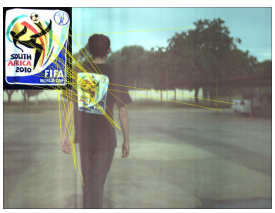
Fifa10 Bounds



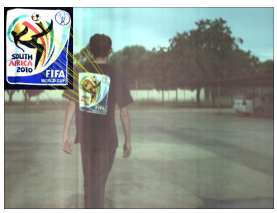
Fifa11 Raw

Fifa11 RANSAC

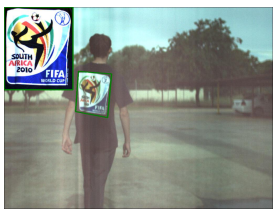
Fifa11 Bounds



Fifa12 Raw



Fifa12 RANSAC



Fifa12 Bounds