**EECE5639**

**COMPUTER VISION**

**PROJECT 2**

**ROHIT RAO NAINENI**

**NU ID- 001738178**

**ABSTRACT**:

This project deals with finding corner features in multiple images and to align the images in mosaic by estimating a homography between corresponding features. An image is warped into the coordinate system of the second one to produce a mosaic containing the union of all pixels in the two images.

**Description of Algorithms**:

1. **Finding Corner Features:**

The sample images used to test our program are not scaled. The algorithm takes two images and finds common features between the two images. The corner detector used is a Harris corner detector to find corner features in the two images. The **input arguments** are:

* 1. Image [m x n] Input image to be processed,
  2. der\_sigma: 1x1 Derivative scale - standard deviation of the Gaussian filter used for pre-smoothing,
  3. int\_sigma: 1x1 Integrative scale - standard deviation of the Gaussian filter used for summing over the pixel neighborhood.
  4. Threshold - 1x1 Corner response function threshold. Values of the corner response function below this threshold are neglected.

The output given by the Harris corner detector function is corner matrix [N x 3]. Each row of 3-column matrix represents one corner point. The first two columns describe the position of the corner of an input image (first column = y coordinate, second column = x coordinate). The third column contains the value of the corner response function.

**Harris detector algorithm:**

1. Firstly, the image is pre-smoothed. We create a 1D Gaussian filter and smooth by convolving in both x and y directions.
2. Compute image derivatives by convolution with kernel [1 0 -1] in both x and y directions.
3. Compute quadratic terms of the derivatives.
4. Compute the sums of the products at each pixel using a window averaging.
5. Compute the corner response function and hard-threshold it - values smaller than threshold are set to zero.
6. The final phase of the Harris detector is non-maximal suppression - we will loop over all points detected so far and only keep those where the corner response function is a local maximum in a 3x3 neighborhood. All other output pixels are set to zero.

The output corners is a list of triples: coordinate values x and y and the value of the corner response function.

1. **Correspondences between the two images:**

Given two set of corners from the two images from step 1, Normalized cross correlation (NCC) of image patches is computed centered at each corner. This is a process and potential corner matches are chosen by finding pair of corners (one from each image) such that they have the highest NCC value.

* 1. A neighborhood window of 20x20 is chosen for a corner point of an image and is compared with the corner point neighborhood of the second image using normalized cross correlation. The function used in MATLAB is NORMXCORR2(TEMPLATE,A) which computes the normalized cross-correlation of matrices TEMPLATE and A. The resulting matrix contains correlation coefficients and its values may range from -1.0 to 1.0. The center element of the resulting matrix is taken since it has zero lag.
  2. If a set of features from a window in the first image was a 80% or greater match to a set of features in the second image (0.8 threshold), a correspondence is generated.

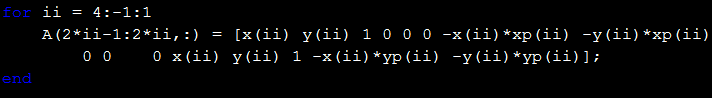
The initial point correspondences is plotted and shown below:

1. **Estimate the homography using the above correspondences**.

The above correspondences are likely to have many errors (outliers).We then use RANSAC to robustly estimate the homography from the noisy correspondences.

Algorithm:

1. Choose four random points at a time which is the minimal number of points needed to estimate a homography.
2. We set h33 = 1 and built the A matrix,



And the B matrix which are just the corner points arranged accordingly in a matrix.

1. Compute a homography estimate from these four points.
2. Map all points using the homagraphy and compute the distances between the predicted points and the observed points i.e. get the norm of the differences. To determine the number of inliers compare the error estimate of points with a threshold and include those points whose error is less than threshold .
3. Compute the **least-squares** homgraphy from all the inliers in the largest set of inliers by updating the Homography and the inliers estimate only if the error is less than the preceeding one.

Find the final homography estimate using all inliers

1. **Warp one image onto the other one**:

The steps are as follows:

1. Determine how big to make the final output image so that it contains the union of all pixels in the two images.
   * + - Firstly, find the source and destination image’s boundaries corresponding to the destination frame.
2. Copy the image that does not have to be warped into the appropriate location in the output.
3. Warp the other image into the output image based on the estimated inverse of homography.
4. Feathering

**Flow Chart:**

1. Unscaled Input Image2

1. Unscaled Input Image1

2. Sigma1 for pre-smoothing,

3. Sigma2 for window averaging

4. Corner response function threshold

2. Sigma1 for pre-smoothing,

3. Sigma2 for window averaging

4. Corner response function threshold

Sparse set of corner features.

Window size = 20x20,

Threshold = 0.8

Set of correspondences between the two images

Np (Number of correspondences)iterations

Threshold for inlier estimate = mean(||||)

Homography estimate using inlier correspondences

Feathered image 2

Feathered image 1

Stitched Image

**Experiments:**

**Values of Parameters used:**

**Observations and Conclusions:**

**Extra Credit:**