Project 1: My AutoPano

Keshubh Sharma, Dushyant Patil

I. Introduction

The purpose of this project is to stitch two or more images in order to create one seamless panorama image. Each image should have few repeated local features (∼ 30-50 % or more, empirically chosen). In this project, you need to capture multiple such images. The following method of stitching images should work for most image sets but you'll need to be creative for working on harder image sets.

II. Phase1: Classical Approach

The objective of Phase 1 was to implement the image stitching on 2 or more images. The image stitching is implemented in following steps:

1) Generate corners using CornerHarris
2) Implement Adaptive Non-Maximal Suppression (ANMS)
3) Create feature descriptors
4) Feature matching
5) Homography generation and RANSAC
6) Image warping and blending

The corners were detected using CornerHarris function which were refined using local maxima and Adaptive non maximal suppression (ANMS)). These detected corners are used to create features and used to match 2 images. The matches are used to get a homography matrix which is used to blend or stitch the images. The in depth processes are as follows:

1) Corner Detection: The images are converted to grayscale and are fed to a **CornerHarris** function which detects corners based on the input parameters such as the Gaussian kernel size, neighborhood size and Corner Harris parameter. The Corner Harris function gives a corner metric score. This score denotes the probability of a point being corner. Using the corner score we detect initial corner points as shown in Fig 1. These corner points are fed to a **ndimage.maximum filter** function to filter out the local maxima of the corners in the neighbourhood of the size specified by user.
2) Adaptive Non-Maximal Suppression (ANMS): The intent of ANMS is to get equally spread corners in the image to avoid artifacts in warping. The output of the corner detection step is a corner score image and nStrong number of corners which act as in input for ANMS. The ANMS algorithm implementation gives the coordinates of the **nBest** corners which are equally spread over the image. The output of ANMS is as shown in Fig 2
3) Feature descriptor: The feature descriptor are vectors which represent the standardized intensity values of the corner and its neighbouring pixels. A 40x40 patch centered around the ANMS output corner is sub-sampled

Fig. 1: CornerHarris output

Fig. 2: ANMS output (dilated)

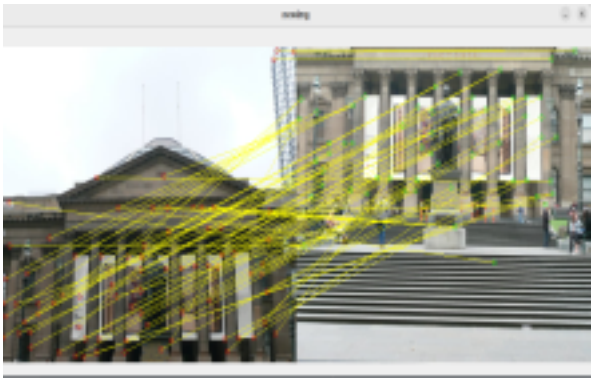

Fig. 3: Matches using feature descriptors

Fig. 4: RANSAC Output

to a 8x8 matrix. a Gaussian filter is applied to this sub-sampled matrix. The matrix is reshaped into a 64x1 vector which acts as a feature descriptor for the center pixel (corner output of ANMS). Feature vectors for all the corners of the image is created using this method. The feature descriptor generation is done for the second input image.

4) Feature matching: Feature matching takes the feature vector set and corner coordinates of both the images. For each corner in first image a sum of squared distances of all the feature vector elements **(SSD)** is done for all the corners in the second image. The SSD output is stored along with corresponding coordinates of corners in both the images. This matrix is sorted around the SSD axis. If the ratio of the first SSD to second SSD is below a threshold (0.8 in our code) the distance and the matching coordinates are stored in a final matches matrix.
5) Draw Matches: The final matches output from Feature matching can be visualized using the match coordinates and a concatenated image of the 2 input images. The visualization shows that there are a few false matches and a few good matches. The output of drawmatches is shown in Fig 3
6) RANSAC and Homography: The output of feature matching is the input for the RANSAC algorithm. The **RANSAC algorithm** is used to find the maximum number of inliers and eliminate false matches. The RANSAC algorithm takes as input, any 4 random matching pairs from both the images. These 4 pairs are used to find a homography matrix which is used on all the matches in both the images to find SSD between them. If the SSD value is less than a user set threshold, the pair is considered an inlier. This process of random pair generation and SSD comparison is repeated for number of iterations specified by user. The output of the SSD is compared with previous iteration to detect maximum number of inliers and their corresponding homography matrix. This set of maximum pairs is used to find the final homography matrix for both the images ($\tilde{H}$)
7) Image warping and blending: The homography matrix is used to find the projected corners of image1 from perspective of image 2. The projected corner points were obtained using the homography equation with 8 H coefficients. cv2.warpPerspective was used to warp the image to align the 2 images. Then a weighted addition was done on the overlapping parts to get average intensity values at the intersection of 2 images.

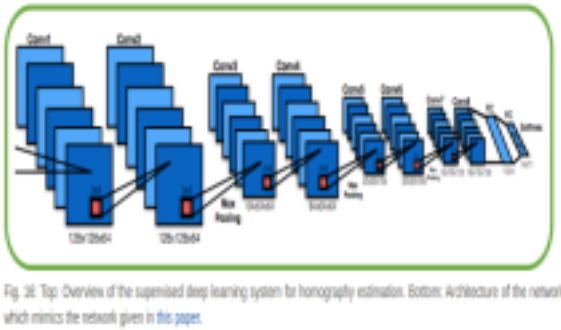

Fig. 5: Supervised Approach

III. Phase2: Deep Learning Approach

The Deep learning approach is to be completed in 2 approaches - (1) Supervised approach (2) Unsupervised approach. The following steps were completed to execute the supervised approach:

1) Data generation: A class 'InputImages' was created to generate labeled dataset. The image was input from dataset and a random corners C_a were generated to create patch P_a of the size 128x128. The corners C_a were randomly perturbed in the image to generate another set of corners C_b. Homography between C_a and C_b was calculated using cv2.getPerspectiveTransform function. The inverse of this homography was created and used to warp the source image I_a to create a warped destination image I_b. A patch corresponding to the initial corners C_a was created in the I_b. This patch P_b is stacked along with patch P_a and their respective 4-Point hmography matrix H-4Pt which is created using C_a-C_b. This creates the labelled dataset.
2) the neural network design: A class 'homographyNet' was created which consists of 8 convolutional layers, 3 Max Pooling layers, and 2 linear layers. The non

linearity is added to the neural network after each layer using 'relu' activation function. The output of last layer was binned into 21 bins using SoftMax. The architecture is shown in Fig. 5

3) Training the neural network: The training dataset consists of 5000 images. The regression model was trained for 'n' number of epochs. The learning rate is to be selected for training the model. The MSELoss function is used as a loss function. An SGD optimizer is used to avoid overfitting/ underfitting with the learning rate 0.01. ReLU activation function is used for convolutional layers and softmax function is used for output layer.