Indian Institute of Technology, Bombay

Foundations of Digital Image Processing
CS 663
(Autumn 2023)

PAGE MERGER

Prepared By:

Tanisha Chawada(23M1071)
EE Department

Chanchal Gupta(23M1069)
EE Department

Problem Statement:

Let's say you urgently wanted to scan a document and send it out over email, but don't have access to a scanner. You can use your mobile phone to take pictures of the document, but if the document is large in size, any one picture might be insufficient to cover the entire document. In such a case, you will need to align the different images you clicked, and blend them to create a meaningful collage. You will also want the image to not be crooked but aligned with the horizontal and vertical axes. Thus,

Given a pair of images I want to stitch them to create a full document.
Note, one fundamental requirement is that both images need to share some common regions.
Also, the solution, need to work even if the pictures have different Scaling or Angle (Perspective) or Spatial position or Capturing devices

Overview:

Image stitching is a method for generating a larger picture by encasing numerous pictures taken from an analogous angle without losing any information.

Feature matching, additionally referred to as image matching, is the procedure of finding similarities between two pictures of the same scene or object. It is an essential part of many computer vision applications, including image registration, camera calibration, and object recognition.

Identifying a list of interest points and associating each one to an image descriptor from the individual image data is a popular method for image matching. Developing some initial feature matches between these images occurs after the features and their descriptors have been obtained from two or more pictures.

Finding Geometrical similarities between matched point to align images and then warping them together.

Solution Approach:

The images are passed into images stitching pipeline which extracts and matches the features in the images, and warps the images together.

The main steps of converting the piecewise images of a document into full image is as follows:

Input: n ordered images

• Extract ORB features from all n images

• Match the features between each pair of images and estimate pairwise Homography matrices using RANSAC.

• Find geometrically consistent Homography for each image.

• Warp each image.

Procedure:

Image Matching

Features matching or generally Image Matching, a part of many computer vision applications such as image registration, camera calibration and object recognition, is the task of establishing correspondences between two images of the same scene/object.

A common approach to image matching consists of detecting a set of interest points - each associated with **i**mage descriptors from the individual image data. Once the features and their descriptors have been extracted from two or more images, the next step is to establish some preliminary feature matches between these images.

Main Feature Descriptor Algorithms

- SIFT(Scale Invariant Feature Transform)
- SURF(Speeded Up Robust Feature)
- BRISK (Binary Robust Invariant Scalable Keypoints)

- BRIEF (Binary Robust Independent Elementary Features)
- ORB (Oriented FAST and Rotated BRIEF)

The ORB algorithm is an alternative to SIFT and SURF. It is 2 times faster than SIFT.

We first created the orb object using open-cv library. Then extracted the keypoints and descriptors for each image and them used brute force feature or descriptor matching algorithm to match the features between two images.

Why we need Descriptors?

Here's two simple examples where only the position and keypoint area will not help us:

- If you have an image A (of a bear on a white background), and another image B, exact copy of A but translated for a few pixels: the extracted keypoints will be the same (on the same part of that bear). Those two images should be recognized as same, or similar.

But, if the only information we have is their position, and that changed because of the translation, you can not compare the images.

- If you have an image A (let's say, of a duck this time), and another image B, exactly the same duck as in A except twice the size: the extracted keypoints will be the same (same parts of the duck). Those are also same (similar) images.

But all their sizes (areas) will be different: all the keypoints from the image B will be twice the size of those from image A.

So, **here come descriptors:** they are the way to compare the keypoints. They summarize, in vector format (of constant length) some characteristics about the keypoints. For example, it could be their intensity in the direction of their most pronounced orientation. It's assigning a numerical description to the area of the image the keypoint refers to.

Brute Force Matcher matches the descriptor of a feature from one image with all other features of another image and returns the match based on some distance calculation. So in another words, given 2 sets of features (from image 1 and image

2), each feature from set 1 is compared against all features from set 2. It is slow since it checks match with all the features.

Removing Outliers

The matched features that we obtained by applying are not all good matches. We have to consider only those matches which are good and help us to find homography transformation matrix. The bad matches are called outliers. For this we use RANSAC .

Random Sample Consensus or RANSAC is an iterative algorithm to fit linear models. Different from other linear regressors, RANSAC is designed to be robust to outliers. Here, I will use RANSAC to estimate the Homography matrix. Noting Homography is very sensitive to the quality of data we pass to it, hence need an algorithm (RANSAC) that can filter irrelevant points from the data distribution.

Geometrical Transformation

The images are taken from same view point but with different angles or rotating camera, so affine transformation will not work here.
We need something that transform one plane into another.
Thus homography come into account.

The homography matrix is a 3x3 matrix but with 8 DoF (degrees of freedom),

A homography is a perspective transformation of a plane, that is, a reprojection of a plane from one camera into a different camera view, subject to change in the translation (position) and rotation (orientation) of the camera.

Perspective transformations map 3-D points onto 2-D image planes using the transformation matrix that incorporates the camera characteristics: focal length, optical centre, and the extrinsic parameters (rotation, translation) .

So, again fundamentally Homography matrix is a mapping between two planes. We have considered it here as a mapping from the image plane to a physical plane, but it could map between two image planes. A homography is a type of projective transformation in that we take advantage of projections to relate two images. Homographies were originally introduced to study shifts in perspective, and they

have enabled people to better understand how images change when we look at them from a different perspective.

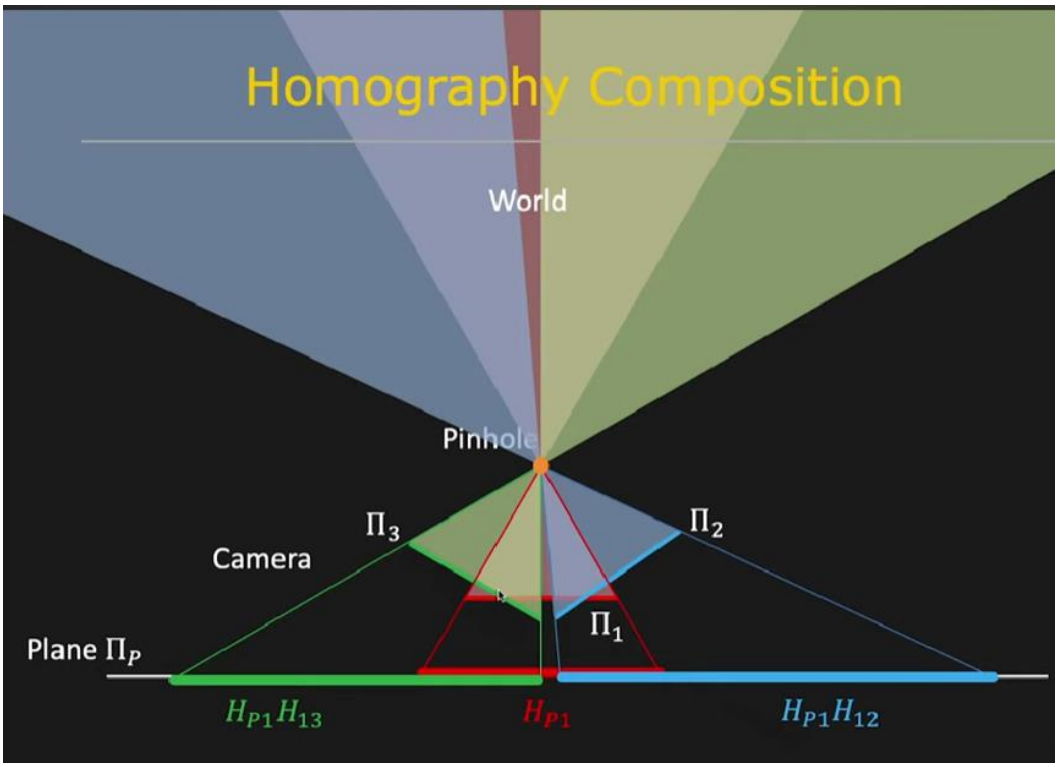

Fig 1.1

Now since a homography is a 3×3 matrix we can write it as

$$H = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix}$$

Let us consider the first set of corresponding points – (x_1, y_1) in the first image and (x_2, y_2) in the second image. Then, the Homography H maps them in the following way

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

To calculate the homography between two images, we must know at least four corresponding points between them. OpenCV robustly estimates a homography that fits all corresponding points in the best possible way. The point correspondences are found by matching features like SIFT or SURF between the images.

The RANSAC algorithm (RANdom SAmple Consensus) is used to estimate a Homography. It computes a homography and gives you a prediction for which pairs are inliers and which are outliers.

Warping

Finally,we can apply the transformation by using the cv2.warpPerspective function. The first parameter is our original image that we want to warp, the second is our transformation matrix M (which will be obtained from homography_stitching), and the final parameter is a tuple, used to indicate the width and height of the output image.

After calculating the transformation matrix (which in this case is the Homography_Matrix), apply the perspective transformation to the entire input image to get the final transformed image.

Results:

1. Image Stitching for 2 images:

First image is of feature matching between two images.

Second is the stitched image.

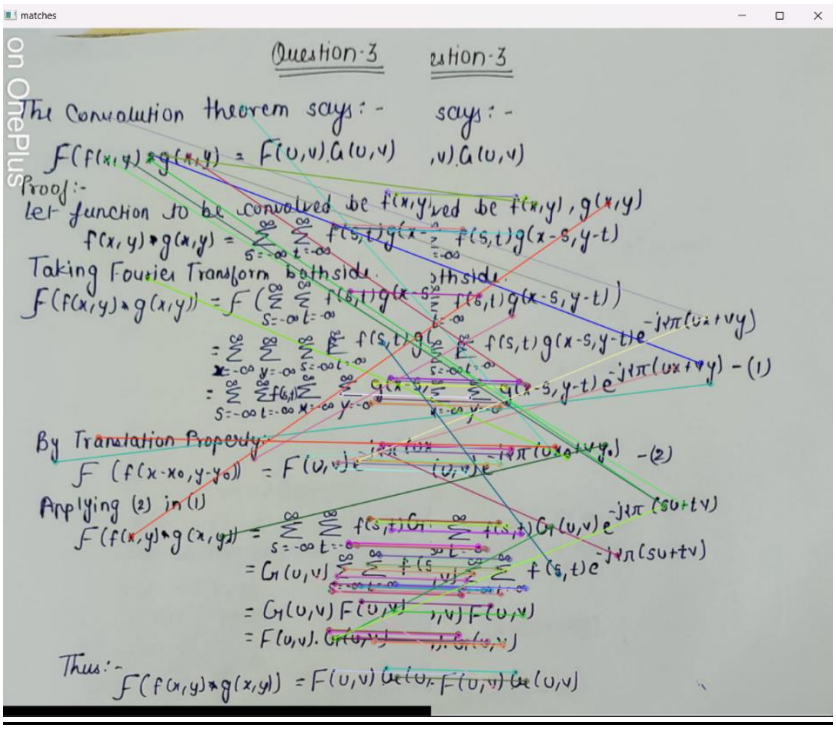

Fig 2. Matches between two images

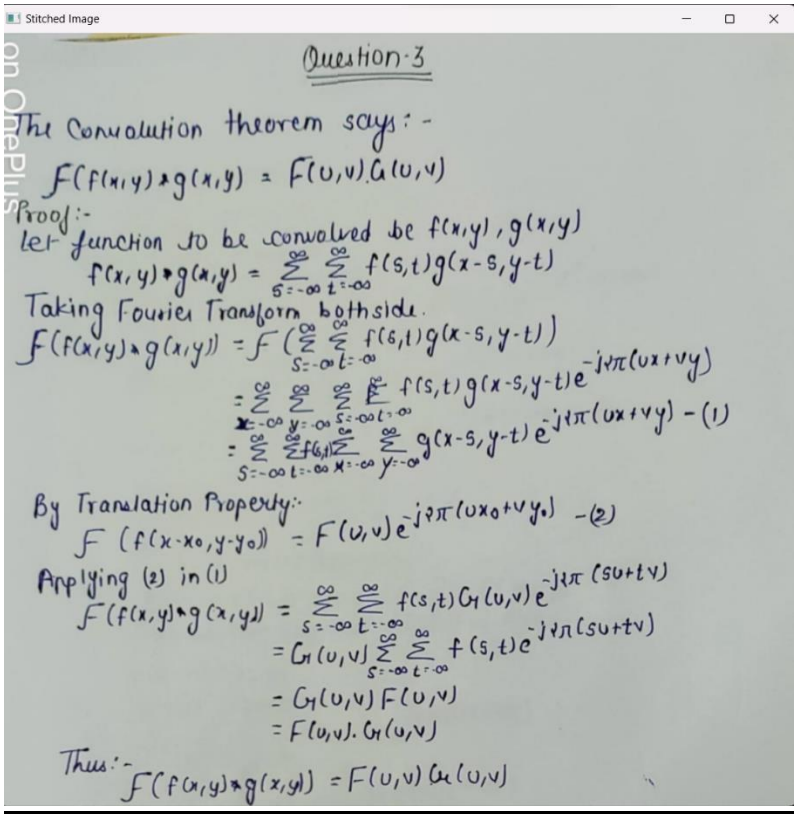

Fig 3. Stitched Image

2. Image Stitching for 4 images:

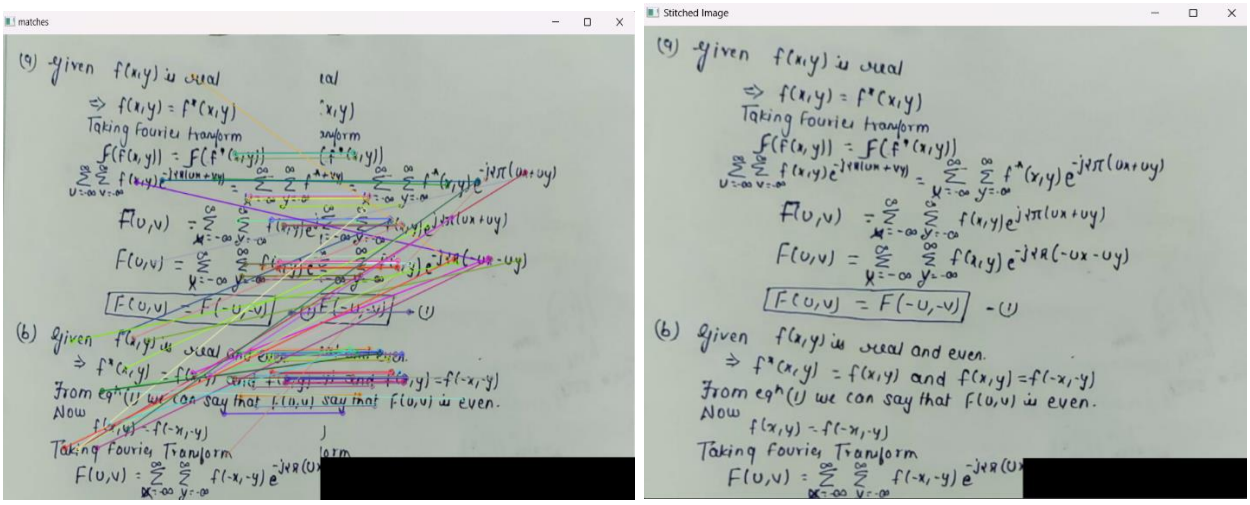

Fig 4. Matches between first two images and stitched image

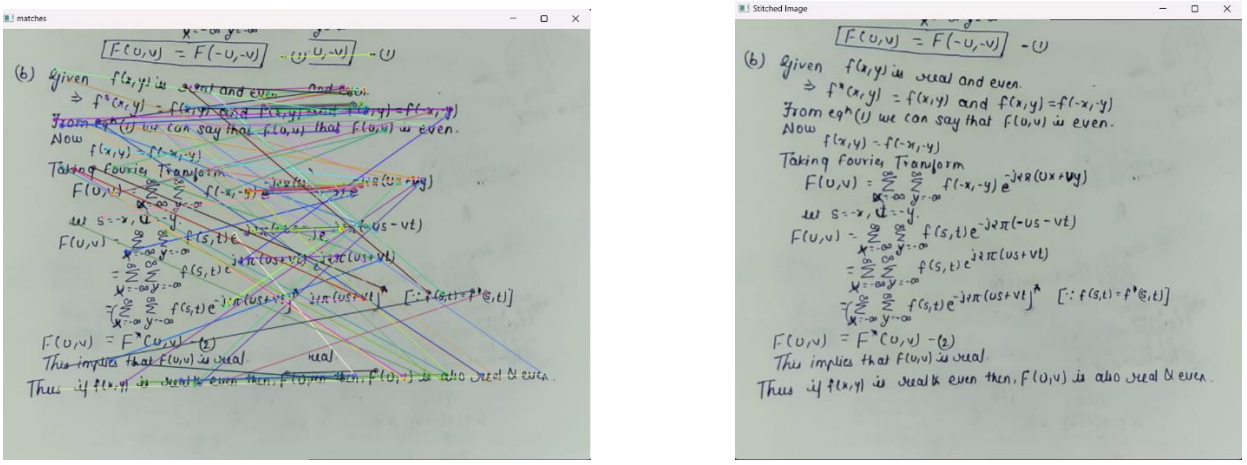

Fig 5. Matches between last two images and stitched image

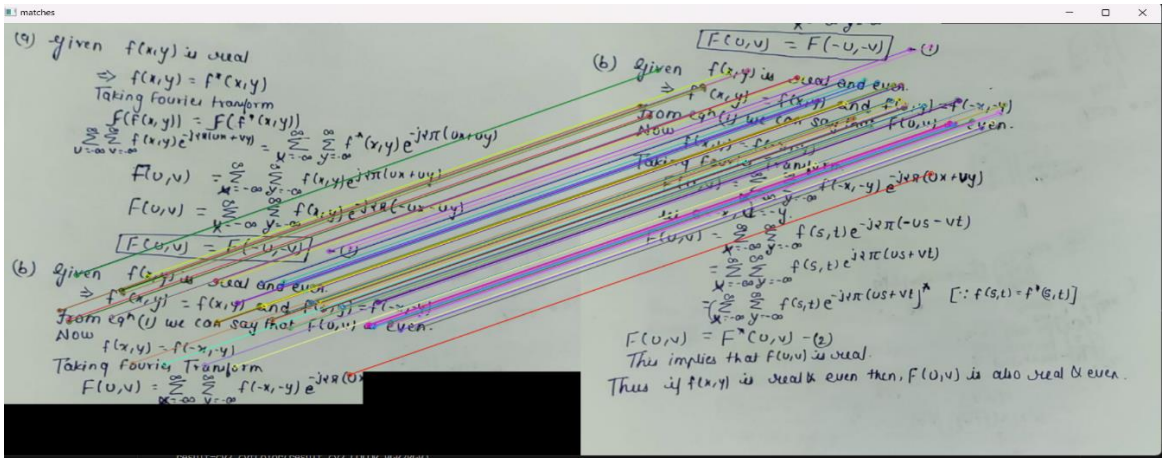

Fig 5. Matches between images obtained by stitching above two images

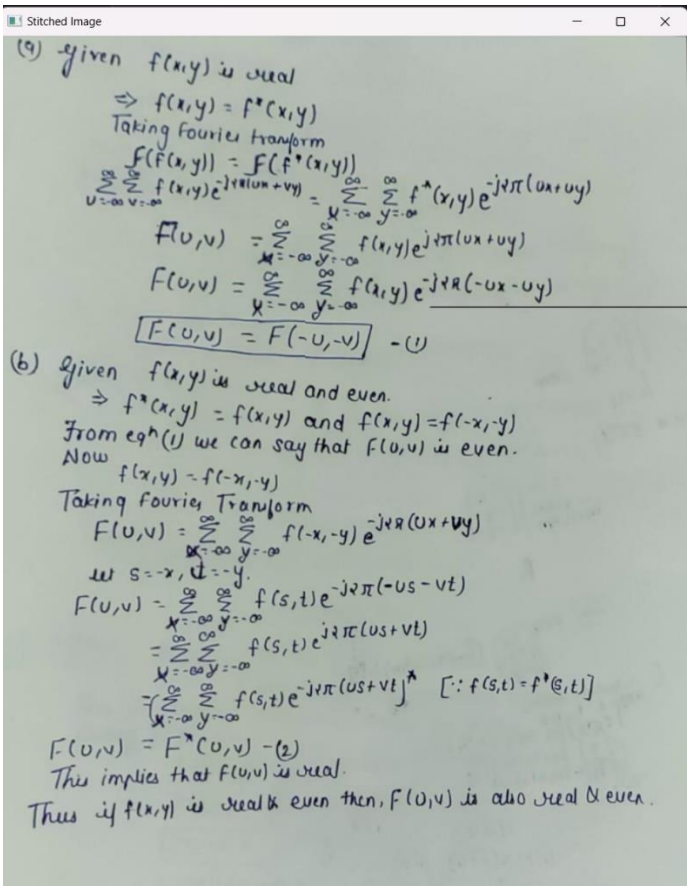

Fig 6. Final Output

Conclusion:

We have used our on data to test the code. It works well with 2,3,4 images but does not perform very well with higher number of images.