Understanding Homography for image stitching

SWB bootcamp on Computer Vision, University of Lagos

Rohan Gupta
cybershiptrooper@gmail.com

1 Introduction

Image stitching has an extensive research literature and several commercial applications. The basic geometry of the problem is well understood, and consists of estimating a 3×3 camera matrix or homography for each image. This estimation process needs an initialisation, which is typically provided by user input to approximately align the images, or a fixed image ordering. However, in reality such images may not be easily obtainable. Our goal is to use popular feature descriptors to extract and match local features which remain the same across two images with a common scene and utilise it for obtaining a stitched image.

In this particular report, we consider the problem of stitching images given the camera has the freedom to rotate around the vertical axis. Since the camera's viewpoint is rotating, the image projections do not lie on the same plane. Hence, one must find appropriate perspective transforms in order to produce a plausible stitched image. The problem can be divided into three parts. Specifically:

- Extracting matching features and ordering images given the matches
- Finding the appropriate perspective transformation between each image and the common final image, and warping each planes appropriately
- Blending the images to produce the final stiched image.

Matching features can be extracted using popular feature descriptors like SIFT, FAST, BRIEF and ORB. Since these descriptors only consider local features, one may get many outliers. To prevent false positives, we use Lowe's ratio.

The problem of finding homography can be solved using a set of four matching features. However, this assumes that the features extracted are not noisy and some inherent idealities of the scene captured itself, i.e., the image is a projection of a scene near infinity, or the camera has neglegible translation. RANSAC algorithm [1] solves this problem, by estimating parameters only using a subset of data identified as inliers. To finally warp the image into the final plane we use inverse warping to prevent aliasing. We constrain ourselves to produce warped images of the same size for implementational ease.

The problem of blending can be solved easily using a variant of alpha composting known as feathering blending. One must however, tune the window size to prevent ghosting effects and sharp edges. I found empirically that a window size of 0.2 of the smaller image usually works well for real examples. For images with a large spectral range(having higher frequency components like grass), one must use pyramid blending.

2 Background Required

As mentioned in the Introduction, problem of image stitching involves domains like feature extraction and matching, perspective transforms, image blending etc. For better understanding of the algorithms and methods used in this paper, reader is expected to have basic understanding of key-points detection, homogeneous image transforms and alpha composting. Few of these ideas and approaches are explained in this paper.

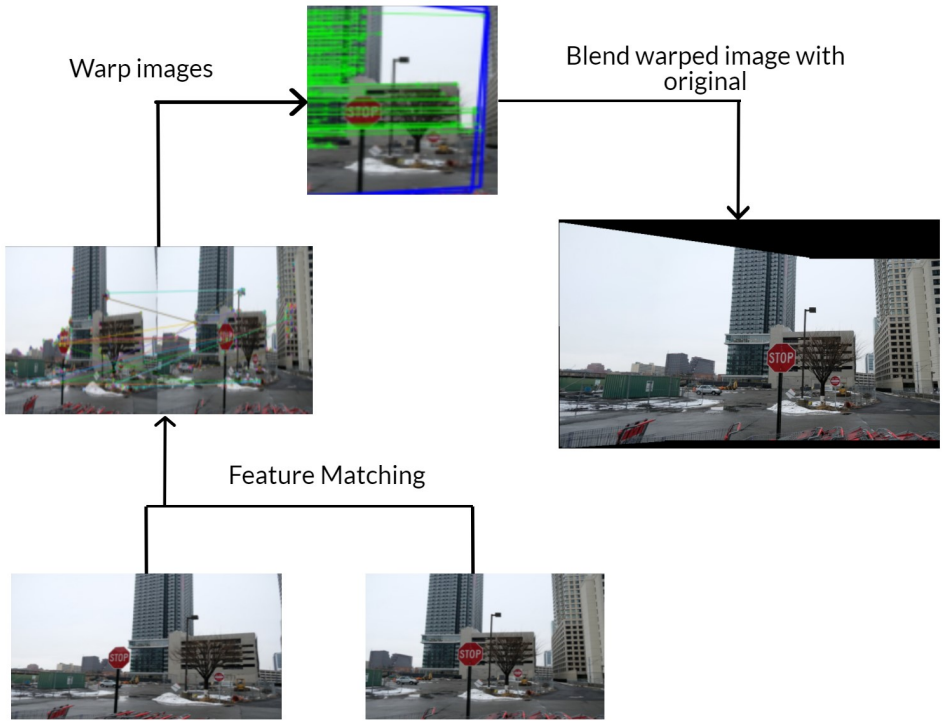

Figure 1: Image stitching pipeline for two images

3 Related Work

Invariant Feature based stitching has been studied thoroughly. [6] provides a nice summary of all the literature in the field. This report looks at a few of these implementations, considering only cases with 1-DOF. More complex models have been studied where more freedom is given to the camera and the scene. Homography fails in such cases and one must use epipolar geometry to account for them [3].

There have been some success in solving image stitching using convolution neural networks [8]. The pipeline can be divided into two parts: Image registration and reconstruction respectively. The problem with CNNs however, is that they fail to capture high pass(local) features as they get deeper. Hence, feature descriptors based on Histogram Oriented Gradients and Hough transforms like SIFT are more popular in the field. Additionally, CNN based methods are slower for lower resolution images. On the flipside, CNNs can encode global locations of the image and hence do not need an additional clustering module to match local features based on their global relative positions. Many deep learning modules [9] [10] have outperformed RANSAC based homography estimators although again, with the cost of computation.

Object centred homography [11] leverage recent advances in object detection and deal with problematic failures when objects are cropped, omitted, or duplicated. Such methods can be used to determine when there is non-recoverable occlusion in the input data.

The models hence can be divided into mainly algorithmic and CNN based methods. This report reviews a few of these architectures.

4 Algorithmic methods

4.1 Formulation

Our pipeline for image stitching is shown in Figure 1 using a simple two image case. The methods used to solve each component are dilineated hereafter.

4.2 Feature Descriptors

Following algorithms helps in key-point detection and associate a feature vector to each one of them.

SIFT :- SIFT algorithm [18] is based on the Difference of Gaussians (DoG) method. It can be divided into 4 basic steps.

- Estimate a scale space extremum using the DoG.
- a key point localization where the key point candidates are localized and refined by eliminating the low contrast points.
- a key point orientation assignment based on local image gradient.
- a descriptor generator to compute the local image descriptor for each key point based on image gradient magnitude and orientation

SURF :- SURF [19] approximates the DoG (used in SIFT) with box filters. Instead of Gaussian averaging the image, squares are used for approximation since the convolution with square is much faster if the integral image is used. Also this can be done in parallel for different scales. The SURF uses a BLOB detector which is based on the Hessian matrix to find the points of interest. For orientation assignment, it uses wavelet responses in both horizontal and vertical directions by applying adequate Gaussian weights.

ORB :- ORB [20] is a fusion of the FAST key point detector and BRIEF descriptor with some modifications. Initially to determine the key points, it uses FAST. Then a Harris corner measure is applied to find top N points. FAST does not compute the orientation and is rotation variant hence not suitable for cases with more than 1 DoF (Degree of Freedom). It computes the intensity weighted centroid of the patch with located corner at center.

BRISK :- The BRISK algorithm is a feature point detection and description algorithm with scale invariance and rotation invariance. It constructs the feature descriptor of the local image through the gray scale relationship of random point pairs in the neighborhood of the local image, and obtains the binary feature descriptor. This is a very poor feature detector.

4.3 Feature Matching

The matching of detected and described features means determining the correspondence between descriptors in images which show the particular object (or scene) in different perspectives. In this paper, matching features between two images is performed by Brute-Force algorithm. It can use Euclidean or Hamming distance, depending on type of obtained descriptors while key-point detection. In order to eliminate outliers from the results, RANSAC is used as explained in further sections. BruteForce matching is usually combined with k-Nearest Neighbors algorithm (also known as kNN) and ratio test.

4.4 Image Alignment

Image alignment comprises of two sub-problems, namely, finding the correct perspective transform between two images and applying the transform to the image.

To understand homography, the problem of finding the correct transform, one must understand the role of homogenous coordinates through in applying perspective and affine transforms on image. Given an image space (x, y), one can easily see the there are no linear transforms of the form

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \mathbf{A} \begin{pmatrix} x \\ y \end{pmatrix}$$

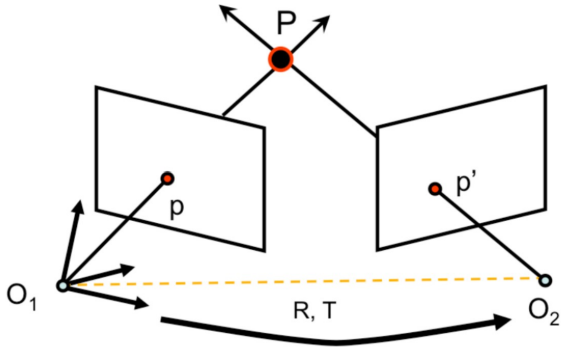

Figure 2: Homography mapping, graphical illustration [3]. The perspective transform is changing the origin(which is nothing but the camera's view vector) and the axes, relative to the image to map the point p to p'

such that $\tilde{x} = x + \Delta_1 x$ and $\tilde{y} = y + \Delta_2 y$. However, when one moves to a third dimension, changing the coordinates to (x, y, w), we can easily see that the transform of the form

$$A_{3x3} = \begin{pmatrix} 1 & 0 & \Delta_1/w \\ 0 & 1 & \Delta_2/w \\ 0 & 0 & 1 \end{pmatrix}$$

does the job. In this space, the image becomes a plane parallel to the x, y plane. Fixing w to one is the popular way to define this plane properly. When returning to the 2-dimensional coordinate system, one must divide do the following:

$$(x, y, w) \rightarrow (x/w, y/w, 1) \rightarrow (x/w, y/w)$$

Consequently, points in the image plane are equivalent to lines passing through origin in the homography space(this can be proved by observing that multiplying the coordinates by a scalar and finally dividing by w leads to the same point). A homography transforms one image plane to the another. For a query point, this mapping can be written as

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

where $(x, y, 1)$ is the point's coordinates in the source image plane and $(\tilde{x}, \tilde{y}, 1)$ is the coordinate in the plane the point is being mapped to(see Figure 2). Since the final coordinate is kept 1, h_{33} must also be kept constant(i.e. 1). The equation can be rewritten as:

$$\begin{pmatrix} x & y & 1 & 0 & 0 & 0 & -x\tilde{x} & x\tilde{y} & -x \\ 0 & 0 & 0 & \tilde{x} & \tilde{y} & 1 & -y\tilde{x} & y\tilde{y} & -y \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{pmatrix} = \mathbf{0}$$

Since we have fixed h_{33}, there are a total of 8 unknowns. Hence, if there are four such matching pairs(two equations each pair), one can find a solution for the h matrix. Note that there always will be a trivial solution to this set of equations and we are not interested in that. Hence we fix $||h||^2$ to one, where h is our homography vector. If we call the matrix set up by four matching pairs as A, the optimisation problem becomes:

$$min_h(||Ah||^2 - \lambda(||h||^2 - 1)$$

equating the gradient w.r.t. h to zero, we get

$$\Delta_h(h^T A^T Ah) - \lambda \Delta_h(hTh) = 0$$

or,

$$A^T Ah = \lambda h$$

which is the eigenvalue problem. The solution hence is the eigenvector of $A^T A$ with the smallest λ, i.e., eigenvalue. Since $A^T A$ is symmetric this solution always exists and in fact is equal to the vector with the smallest singular value in A's singular value decomposition [4]. One may use more points than required to make it an overdetermined system of equations(note that the solution still exists as the above arguments hold for any A).

To deal with the errors of feature descriptors and the assumptions about the camera and scene stated before, we use RANSAC [1] regression to fit the homography to all the features matched before. The algorithm is very robust to outliers and hence, suited for our application. For any homography the 'outliers' are defined as

$$\mathbf{1}(||Hp - \tilde{p}|| \geq \epsilon)$$

where $(p, \tilde{p})$ form the matching pairs.

The image is then warped into the reference plane using the found homography. We use inverse warping [5] to prevent aliasing in the transformed image. Instead of applying

i $p' \rightarrow Hp$

ii $warped(p') \rightarrow original(p)$

we use

$$warped(p) \rightarrow original(H^{-1}p)$$

to sample in the warped domain instead. Figure below shows the results of warping on a custom avatar.

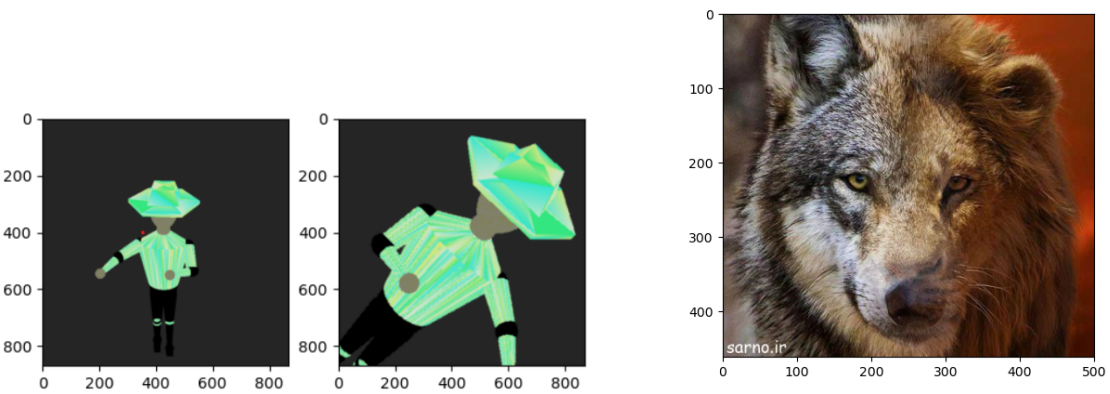

Figure 3: Result of an affine transformation comprising translation of origin to center of image, rotation by 45^o followed by scaling by 2 using inverse transform(a) Blending two images using feathering (b)

4.5 Image Blending

Simple alpha blending was used for this purpose. Figure above shows the results of blending two images using this method. The window size was tuned emperically and works well with most real images. When the images are stacked horizontally, the fraction of each image to be blended is controlled by the window size. Given *img1* and *img2* as the left and right images respectively, and the window size, a simple blending

output can be formally defined as follows:

$$img(x,y) = \begin{cases} img1(x,y), & \text{if } x \leq start \\ img2(x-end,y), & x \geq end \\ img1(x,y)\frac{(end-x)}{\text{window size}} & \\ +img2(x-end,y)\frac{x-start}{\text{window size}} & otherwise \end{cases}$$

where end and start are width of *img1* and end-window size respectively. The window size is chosen as the size of the common scene between the query images (which is already known from feature matchers). However, the function implemented used a further smaller blending window inside the window size where the blending is performed. This was to reduce ghosting edges while blending images with high frequencies or for cases where window size becomes too large. On the flipside, if blending window is chosen to be too small, the edge between images become highly prominent.

References

[1] Fischler et. al. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography

[2] First Principles of Computer Vision. Image stitching. [link]

[3] Kenji Hata and Silvio Savarese. CS231A Course Notes 3: Epipolar Geometry, Stanford University.

[4] Reza Bagheri. Understanding Singular Value Decomposition and its Application in Data Science

[5] CA GLASBEY. A review of image-warping methods.

[6] R. Szeliski. Image alignment and stitching: A tutorial. Technical Report MSR-TR-2004-92, Microsoft Research, December 2004.

[7] OpenCV Feature Matching [link]

[8] Z. Yang and T. Dan and Y. Yang. Multi-temporal Remote Sensing Image Registration Using Deep Convolutional Features

[9] Nie et. al. Learning Edge-Preserved Image Stitching from Large-Baseline Deep Homography

[10] Daniel DeTone, Tomasz Malisiewicz, Andrew Rabinovich. Deep Image Homography Estimation

[11] C. Herrmann, C. Wang, R.S. Bowen, E. Keyder and R. Zabih. Object-centered image stitching

[12] Lang Nie, Chunyu Lin, Kang Liao, Shuaicheng Liu, Unsupervised Deep Image Stitching: Reconstructing Stitched Features to Images

[13] ZHUOQIAN YANG, TINGTING DAN2, AND YANG YANG, Multi-Temporal Remote Sensing Image

[14] Lang Nie, Chunyu Lin, Member, IEEE, Kang Liao. Registration Using Deep Convolutional Features Learning Edge-Preserved Image Stitching from Large-Baseline Deep Homography

[15] Adobe panorama dataset [link]

[16] UDIS dataset https://paperswithcode.com/paper/udis-unsupervised-discovery-of-bias-in-deep

[17] Matthew Brown and David G. Lowe. Automatic Panoramic Image Stitching using Invariant Features.

[18] https://medium.com/data-breach/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40

[19] https://medium.com/data-breach/introduction-to-surf-speeded-up-robust-features-c7396d6e7c4e

[20] https://medium.com/data-breach/introduction-to-orb-oriented-fast-and-rotated-brief-4220e8ec40cf

[21] J. Davis. Mosaics of scenes with moving objects.