Image Stitching

Adrian Andersson & Johan Forslund

1 Introduction

One of the most successful algorithms in computer vision is image stitching. Image stitching is a process where multiple images are overlapped and "stitched" together to create one single coherent image. In some scenarios one camera is not enough to cover a large area. In those cases multiple cameras can work together and the process of image stitching can be used to produce one single image containing the areas the each individual camera has photographed.

The field is relatively old in image processing and has known uses in for example satellite imagery, document mosaicing and medical imaging. The algorithms to stitch images together is incorporated in most of today's cameras and cellphones.

In this report we explore how image stitching can be used in sports to improve the viewing experience, or to make technical decisions. This is done by combining images of multiple cameras to create a larger field of view. The sport and application in question is tennis where two cameras are aligned to capture and film each half of the tennis court with some overlap.

OpenCV provides image stitching functions for python scripting but these are avoided to some extension that is reasonable for the scope of this project.

2 Background

The process of stitching images together is a well researched area in image processing. There are several different techniques to complete this task, where the best technique to use depends on the specific use case. In general terms, the problem is solved by finding geometric transformations between a set of images so that that features in the images overlap. Figure 1 shows an example where three images has been combined to create this panorama effect. Since each image is a plane, the sought after transformation becomes a homography, which can be described as a $3x3$ matrix with eight degrees of freedom. The next step is aligning the images, this can be achieved with a direct or a feature based alignment process. The direct alignment process compares pixels in the overlapping areas of images to find a match while the feature based alignment process uses features from those areas. Usually the feature based approach is preferred as it is more robust [4].

Additional steps might include color- and exposure correction, this is needed if the images are captured at different times or with different cameras, see Figure 1. With knowledge of the cameras' positioning the process can be simplified and some steps can be left out [4].

As mentioned, image stitching can be used to create panorama images. This is desired in certain sports broadcasting to capture a large portion of the gameplay. To accomplish this, two ore more cameras can be setup to capture the game field with different angles, and then the images can be overlapped into a panorama with image stitching, hence creating a broad overview of the game.

3 Method

This section describes the implementation of image stitching using two images. At the end, an extension to this algorithm is added to allow for image stitching on a video sequence.

- *Adrian Andersson, student at Linköping University, Sweden.*
  -mail: adran117@student.liu.se.

  Johan Forslund, student at Linköping University, Sweden.
  -mail: johfo522@student.liu.se.

Fig. 1. Three images are combined into a panorama.

3.1 Camera setup

To simplify the process, it is important to capture the images from approximately the same position. Furthermore, the images should have an overlapping area so that common features can be found between them. To fulfill these requirements, we built a simple rig where two smartphones could be placed in a static position with slightly different camera angles. Figure 2 illustrates how the two smartphones are placed in a cardboard box with dotted lines to visualize the camera angle. The smartphones are positioned so that about 10-20% of one image are overlapping with the other.

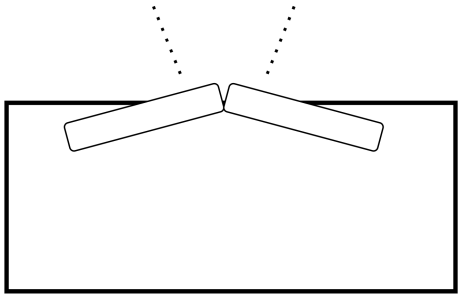

Fig. 2. Illustration of camera setup.

The reason for using two smartphones is to be able to capture video and creating a panorama where the images are synchronized. It is favourable if the two smartphone cameras are identical to avoid distortions due to different camera parameters. However, we were not able to use two identical smartphones in this project.

3.2 Finding and matching keypoints

The following sections describes how keypoints are found, matched, and later on ultimately used to estimate a homography.

3.2.1 SIFT

OpenCV provides a method of finding keypoints with a scale invariant feature transform, more commonly referred to as *SIFT*, method. The method will compute and return valid points of interest in an image. The method is used to generate descriptors in both images that can be used to match the images in the overlapping areas.

3.2.2 Matching keypoints

To match the keypoints generated by SIFT a *K-Nearest-Neighbour* algorithm is used. The algorithm will find the k best matches between descriptors by looking at the euclidean distance in the descriptors's feature space. In order to eliminate false matches a ratio test is used. The ratio test will reject any match of keypoints where the distance between the closest match is greater than half the distance between the second closest match, i.e. it must fulfill Equation 1. The value of $k = 2$ is used throughout the project.

$$d_1 = \frac{d_2}{2} \tag{1}$$

In Equation 1 the distance between the closest match is denoted d_1 and the distance between the second closest match is denoted d_2. An illustration of valid matches are presented in Figure 3, note that the figure only visualizes a fraction of all matches to not clutter the image.

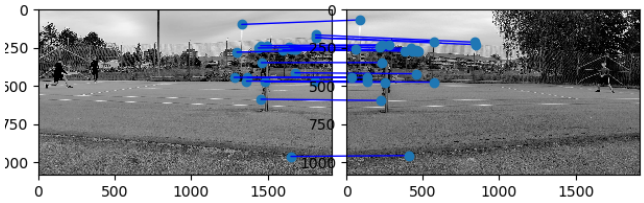

Fig. 3. A fraction of valid matches between the two images using $k = 2$ and the ratio testing in Equation 1.

3.3 Finding the transformation

Given a set of corresponding points from the two images it is possible to estimate a transformation that will align the points, which naturally creates the desired panorama effect. Since this will be a 2D pixel transformation between two images (planes), the images are related by a *homography*.

3.3.1 Homography estimation

After the keypoint matching step, we have multiple point correspondences $u_i \leftrightarrow u_i'$. We want to relate each correspondence with a homography H according to Equation (2).

$$Hu_i = u_i' \tag{2}$$

Since homographies are defined in projective space [2], the points will have homogeneous coordinates $(u, v, 1)$ and the homography will be a 3x3 matrix. By expanding Equation (2) we can rewrite it as Equation (3).

$$\begin{bmatrix} 0 & 0 & 0 & -u & -v & -1 & v'u & v'v & v' \\ u & v & 1 & 0 & 0 & 0 & -u'u & -u'v & -u' \\ -v'u & -v'v & -v' & u'u & u'v & u' & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} h1 \\ h2 \\ h3 \\ h4 \\ h5 \\ h6 \\ h7 \\ h8 \\ h9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3}$$

We shorten Equation (3) as $A\mathbf{h} = \mathbf{0}$ where $\mathbf{h}$ are the components of H. Furthermore, since column three of A is a linear combination of column one and two, we only have two columns that contribute to solving the equation. Hence every correspondence $u_i \leftrightarrow u_i'$ contribute with 2 equations. Since H is homogeneous, it is enough to solve for eight of the nine values in the matrix, leaving a degree of freedom for scale. Putting all this together, we need four correspondences to solve for the eight unknowns of H. This gives Equation (4) which is similar to Equation (3) except that we use four correspondences instead of just one.

$$\begin{bmatrix} 0 & 0 & 0 & -u_1 & -v_1 & -1 & v_1'u_1 & v_1'v_1 & v_1' \\ u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1'u_1 & -u_1'v_1 & -u_1' \\ 0 & 0 & 0 & -u_2 & -v_2 & -1 & v_2'u_2 & v_2'v_2 & v_2' \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -u_2'u_2 & -u_2'v_2 & -u_2' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} h1 \\ h2 \\ h3 \\ h4 \\ h5 \\ h6 \\ h7 \\ h8 \\ h9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4}$$

We solve Equation (4) by using *singular value decomposition* (SVD), see Equation (5).

$$A = USV^T \tag{5}$$

The solution $\mathbf{h}$ is given as the last column of V, and we can then reshape $\mathbf{h}$ into H, giving us the homography [3].

3.3.2 RANSAC

The solution H in Section 3.3 is found by only using 4 corresponding points. If any of these points are wrong it ruins the homography. To fix this instability, we incorporate the *RANSAC* algorithm.

RANSAC is an interative method to estimate parameters of a model from a set of observed points that contain outliers. This is done by iteratively sampling a small set of data points and determining how well these points fit a solution for the full dataset. When this is done enough times, a robust solution is found by choosing the solution with least amount of outliers.

In the scenario of homography estimation, we sample 4 corresponding points a time to find a homography. Next, this homography is applied to each point u_k and we calculate the euclidean distance to the corresponding point u_k'. Ideally, we want many of those distances to be small since this means we have found a good H. For each correspondence where the distance is larger than a threshold, we consider this an outlier. After repeating the process for many iterations, we choose the homography H that gives the least amount of outliers.

3.3.3 Normalized DLT

The method in which we find $\mathbf{h}$ from SVD as described in Section 3.3 is called *direct linear transform*. This method performs better when all terms of A has a similar scale [1] which can be achieved by using *normalized direct linear transform*.

To apply normalized DLT, we first normalize the set of points u_i and u_i' with a similarity matrix T that translates the centroid to the origin and scales such that the average distance from the origin is $\sqrt{2}$. This matrix T is defined as

$$s \begin{bmatrix} 1 & 0 & -\hat{u} \\ 0 & 1 & -\hat{v} \\ 0 & 0 & 1/s \end{bmatrix} \tag{6}$$

where $\hat{u}$ and $\hat{v}$ are the mean values of respective vector. We then apply regular DLT to obtain $\hat{H}$, followed by a normalization step

$$H = T'^{-1}\hat{H}T \tag{7}$$

to get the final homography.

3.4 Post processing

After the two images are transformed to a single plane the seam is visible. To create a coherent panorama image the seam needs to be removed. This can be achieved by using a transparency mask and fade one image out while the other image fades in in the overlapping areas. This creates a smooth transition between the images, however it also leads to some ghosting effects.

Attempts at both color transfer and histogram matching were made to create a coherent panorama image with matching colors. However both attempts led to poor visual results and the attempts were abandoned as the blending technique achieved the desired results.

3.5 Extending to video

The goal of the project was to apply image stitching to a video sequence. Since the cameras were setup in a static position it was possible to only calculate the homography for the first frame and then applying the same homography to all successive frames. The only manual work of the whole algorithm is to temporally synchronize the videos.

4 Results

Here we present the results from our image stitching algorithm. All images have been captured side-by-side with two smartphones: *OnePlus 5* and *OnePlus 6*. The resolution of the images are $1920x1080$.

Fig. 4. The resulting panorama image after color transfer is applied.

Figure 4 presents the poor visual results after an attempt at color transfer was made. Figure 5 shows the corresponding results after attempting histogram matching. Neither of Figure 4 and Figure 5 presents acceptable results and both methods were thus discarded.

Fig. 5. The resulting panorama image after histogram matching is applied.

The final results following 10000 iterations of our RANSAC algorithm is presented below in Figure 6, and Figure 7. Figure 6 has not had any color correction or blending applied and is thus the result of mere transformation.

Fig. 6. The final result without blending and with 10000 iterations of RANSAC.

Algorithm	Execution time (ms)
OpenCV BFMatcher KNN	25443.1199
Our KNN	405293.7212
OpenCV RANSAC	7.7457
Our RANSAC	6111.5109

Table 1. Execution time comparison between OpenCV functions and own implemented counterparts using all initially discovered keypoints and 10000 iterations of RANSAC.

The seam in Figure 7 is much less noticeable than in 6. However if one would take a closer look there is some ghosting effects during the transition of the two images (take a look at the street light in the middle of Figure 7). The ghosting effects is especially noticeable in the video output as a man walks through the seam with his dog. The ghosting effect occurs as the images are overlapped and not fully opaque, creating a blend of both images with some slight offset to details.

Fig. 7. The final result with blending and with 10000 iterations of RANSAC and linear interpolation between the two images.

In Table 1 it is presented that the OpenCV functions are magnitudes faster than our implemented counterparts. For comparison, OpenCV's brute force matcher's KNN is ≈ 16 times faster than our KNN algorithm and OpenCV's RANSAC algorithm is ≈ 780 times faster than our counterpart under the given circumstances.

5 Conclusions & Discussion

Within the scope of this project image stitching has been performed. Two still images were used to estimate a homography that is then later used to transform a video sequence to a panorama. There has been some experimentation regarding different threshold values and number of iterations of the various algorithms used, however most effort was invested to implementing image stitching with minimal amount of pre-existing code. The SIFT algorithm was taken from OpenCV while

KNN, ratio test, homography estimation, RANSAC, color correction, and blending were done from scratch. While implementing our own functions the outputted images were visually compared to the resulting output from OpenCV's image stitching functions. We believe that our implementation visually matches that of OpenCV with the disadvantage of being much slower. We have tried to vectorize as much as possible in our implementation but it is still hard to compete with the performance of OpenCV that uses multithreading and other advanced performance boosts.

Before the blending technique was attempted, a try at histogram matching and color transfer was made, however with poor results. Both techniques tries to match the colors in one image and apply it to another. This is important as the video and images are captured with different cameras. We believe that the poor result might be because the sun is apparent in one of the images which skews the histogram towards more bright colors, hence creating a visible seam. However, by using a blending technique the slight difference in color was not noticeable.

The final result shows that there are some geometric distortion that elevates to the outer edges. This makes the player to right look larger than the players to the left. The same effect occurs both with our algorithm and OpenCV's. We do not know for certain what causes this, but it could be from the fact that we use different cameras which have slightly different intrinsic parameters.

Further work might include a more sophisticated blending algorithm in order to remove the ghosting effect that is apparent in Figure 7. It would also be preferable to implement some algorithm to automatically synchronize the videos in time since this had to be done manually. Alternatively, this could be solved by triggering the smartphones to start recording at the exact same time. As the videos contain fast motions, such as the moving tennis ball, it is desirable to capture the images at a higher frame rate than what is used. This would reduce the amount of "stuttering" the ball displays because it travels further between frames at lower frame rates.

Image stitching is a very useful technique to create larger images or videos from multiple sources. This could prove very useful in sports events such as football where the game and players are studied intensively and the field is larger than most cameras can capture at once.

REFERENCES

[1] R. Q. Feitosa. 2d homographies.

[2] K. Nordberg. *Introduction to Representations and Estimation in Geometry*. 2018.

[3] T. Opsahl. Estimating homographies from feature correspondences.

[4] R. Szeliski. *Computer Vision: Algorithms and Applications*. 2010.