Computer Vision: Image Alignment and Stitching

Arsen Sheverdin, Mátyás Schubert, Ard Snijders

November 23, 2020

Introduction

In order to efficiently find the correspondence between elements of an initial image and a reference image, so called *Image alignment* algorithms are used. The reference image is usally a transformed version of the initial image: for example, the reference image can be a tilted or re-scaled version of the original image. Image alignment algorithms, in turn, compute these transformations, allowing us to match features of the original image to those of the transformed one. *Stitching algorithms* use the alignment estimation produced by alignment algorithms and merge images into a combined one, aiming at solving problems such as image blurring, differences in image exposure and parallax effects (which results from seeing the same image from different angles). Both of the classes of algorithms are widely used for applications such as video stabilization, photo and video summarization and panoramic image generation.[1]

Image Alignment

Image alignment means transforming an image to match a reference image. This reference image contains mostly the same scene but in different scale, orientation or in other transformations applied. Our goal is to recover these transformations. In order to find out how an image can be transformed to match the reference, we first need to find the similarities between the two images which we can then compare. We are searching for these similarities between keypoints (or features) of the two features as these can be recognized regardless of the transformations. We detect keypoints using SIFT[2] and match them with the built-in brute force matcher of opencv[3]. Results of such found matches can be seen on figure 1. Matches found at this point are

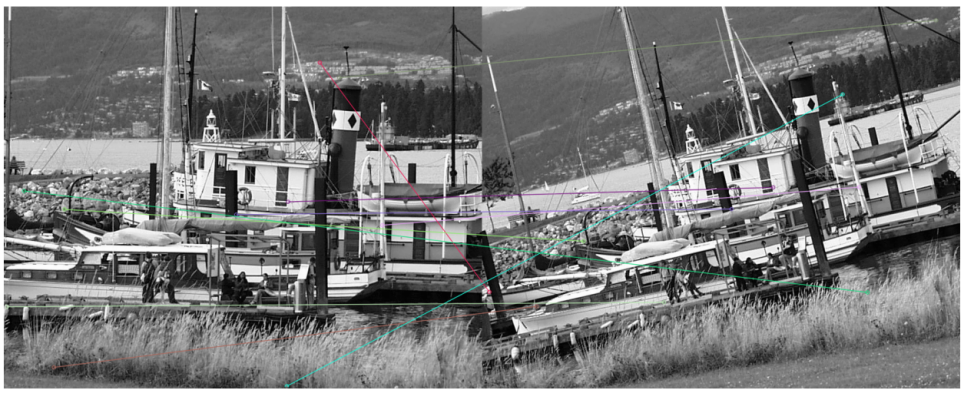

Figure 1: 10 out of 8850 matches detected between boat1.pgm and boat2.pgm

not always accurate. It is possible that certain points get matched with points which do not correspond to the same object/position on the reference image. In order to find the proper transformation while also dealing

with the noise in the matches, we perform RANSAC [4] to estimate the transformation parameters. As there are 6 parameters which we need to recover, we need 6 equations to be able to solve each of them. As a point results in two equations for the linear system, we select 3 random points in each iteration while performing RANSAC. Increasing the number of matches selected would make the linear system over-defined and also increases the chance of an inaccurate match being part of the computation of transformation parameters.

Increasing the number of iterations however often produces better results simply because we give the algorithm more chances to find the optimal solution. We performed a simple experiment where we measured the average maximum inliner count after each iteration while performing ransac. Results can be seen on figure 2. These averages were computed by running RANSAC with 50 iterations over 60 times. We can see that there are no drawbacks from increasing the number of iterations other than of course computation time. Computing an exact optimal number of iterations which gives good results with a high probability from the number of matches is impossible as we would need information on the quality of matches too. After acquiring

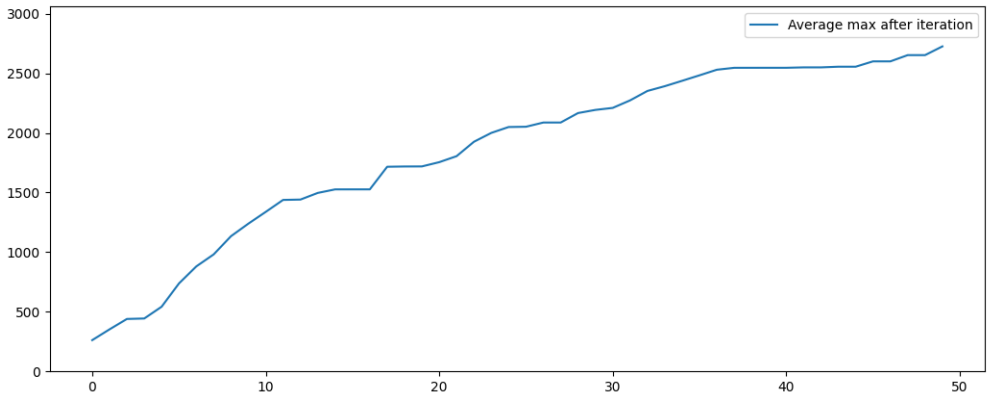

Figure 2: Average maximum inliner count after each iteration

the best transformation based on the number of inliners, we can finally transform the image and compare it with the reference. We use forward warping to transform the image which means some point positions might end up having fractional values. We simply round these values which results in holes in the transformed image. We could use more precise warping methods or we could fill up these holes after the transformation is done. However for the sake of simplicity we simply ignore them as the resulting transformed image is good enough to demonstrate the aligning algorithm we implemented. Alignment of the boat pictures can be seen on figure 3 and 4.

Figure 3: Alignment of boat1.pgm with boat2.pgm as reference

Figure 4: Alignment of boat2.pgm with boat1.pgm as reference

Image Stitching

Image stitching or sometimes called photo stitching is a technique aimed at combination of several different images, which have overlapping fields of view, resulting in one image, having wider field of view or higher resolution. The task of image stitching has wide-spread use ranging from commercial digital camera applications to medical imaging. In this report we discuss a simple method to stitch images together.

Transforming images

The process of stitching can be divided in several steps. The first step, as was discussed in the previous section, concerns estimating the transformation between two images. After this transformation matrix has been obtained, we can use it to project one of the images to be in the same coordinate space as the other, after which the images can be composed together. For our stitching algorithm, we have chosen to project the 'left' image into the space of the 'right' image by transforming the 'left' picture with the transformation matrix. Alternatively, we could also project the 'right' image with the inverse of the transformation matrix to project it 'back' into the coordinate space of the 'left' image. Note that these operations are essentially equivalent; the difference being that with the first method, the 'left' image will be transformed with respect to the right image, and vice versa for the second method. For our stitching algorithm, we perform planar projection, such that the final composition lies on a flat plane. Before we transform the left image, we first separately 'paste' both images on larger black 'canvas' (an array filled with zeros); this ensures that the image is not cut-off by the window frame post-transformation (e.g. due to the horizontal translation from left to right).

Blending

After ensuring that both images are in the same coordinate space, they can be composed together by simply adding the matrices of both pictures together. However, this would produce visible 'seams' in the region where the images overlap. In order to achieve a more natural composition, we would ideally want a smooth transition between images. The task of obtaining a natural-looking stitch between images is called image blending. A variety of techniques are available to blend together images, ranging from a basic gradual blur to more sophisticated techniques such as Laplacian pyramids. We explore two simple ways of blending images; via a linear blur, and with a sigmoidal blur.

Before blurring is applied, we first estimate the degree of overlap between the two source images. In our problem setting, we're only concerned with stitching together 'left' and 'right' images. Resultantly, we choose to only consider overlap along the horizontal axis. The horizontal overlap distance can then be calculated as the translation in x as obtained from the transformation matrix, relative to the length of the image. After obtaining this distance, a gradual blur is applied to the rightmost and leftmost overlap regions for the right and left images, respectively, where the width of the region of interest is determined by the

horizontal overlap distance. A linear blur effect can then be achieved via element-wise multiplication of the rows of the overlap regions with a `np.linspace` vector of equal length, containing values ranging from 1 to 0, and 0 to 1, for the left and right images, respectively. This yields a linear decrease in pixel intensity for the overlap region, for both images. Alternatively, a blur with a sigmoidal curvature can be performed by passing the aforementioned vector through a sigmoid function, yielding a more aggressive transition effect. Note that we apply blur to both images *after* computing the transformation matrix (as the translation for x is required to isolate the overlap region), but *before* transforming one image to the coordinate space of the other. After blurs have been applied, the images can be stitched together by adding their matrices together. An example of a stitched image can be seen in figure 5.

Figure 5: Left: Original images *left.png* and *right.png*. Right: stitched images blended with sigmoidal blur. A total of 75 inliers were found by applying *RANSAC* on the original images.

Provided that two images have a moderate amount of overlap, a projection of adequate quality can be achieved. As can be seen in figure 5, transforming the left image with the transformation matrix yields a good alignment between images. Thus, assuming the transformation matrix is accurate enough, the quality of the stitched image largely hinges on how well the two images blend together. It can be interesting to consider how different types of blurs affect the end result. For instance, consider the result in figure 6, where both source images had a significant overlap, and in which a linear blur was applied on the overlapping region for both images. We can note several things. First, due to the comparatively wide overlap, the degree of rotation between the left and right image, and the linearly decreasing brightness arising from the blur, a so-called 'ghosting' effect occurs (particularly noticeable on e.g. the ceiling in the stitched image). Furthermore, a slightly dark hue can be observed across the overlapping region, as the blur on both images leads to reduced brightness on the overlapping region.

We can attempt to lessen the ghosting effect by applying a blur with a more aggressive contour. Specifically, we can transform the linearly spaced vector by passing it through a sigmoid function. As can be observed in figure 7, this leads to a reduction in ghosting, but it also introduces a darker patch across the overlapping region, which arises from the starker brightness transition in the sigmoidal blur applied to both images. An opposite effect can also occur: in some cases, applying a linear blur can result in a lighter patch across the overlap region: here, the increased brightness from addition of pixels outweighs the reduction in brightness as caused by the blurring (figure 8).

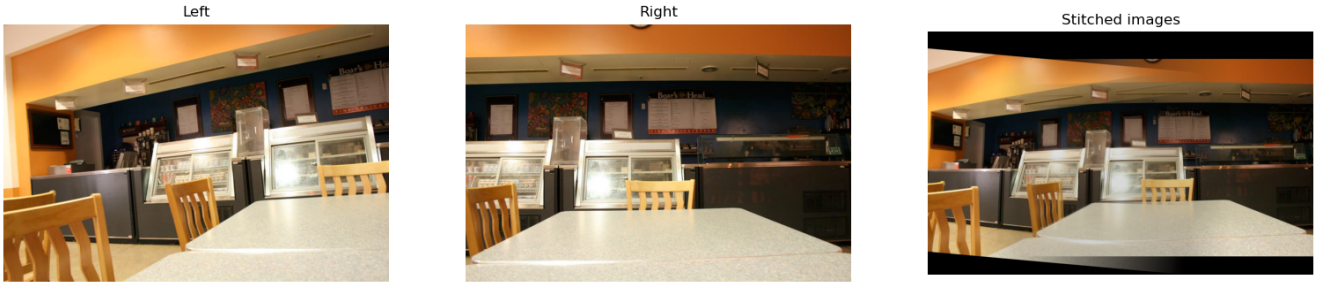

Figure 6: Applying a linear blur leads to a visible ghosting effect, as well as a more subtle dark cast along the region of overlap of the source images.

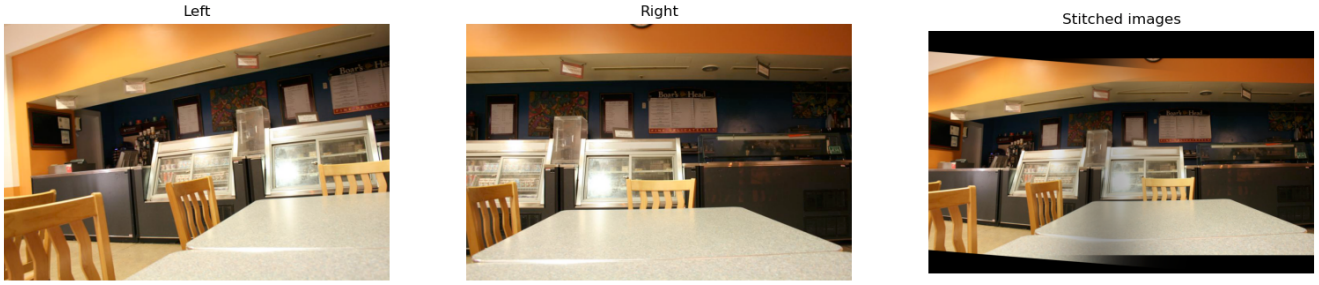

Figure 7: Applying a sigmoid blur leads to a reduction of the ghosting effect, albeit at the cost of a visibly darker cast along the overlapping region.

Figure 8: Applying a linear blur can also lead to increased brightness in the overlapping region (the effect of which, in this case, is actually quite pleasing).

Conclusion

In this report we have shown that with relatively simple techniques, we are able to stitch two separate images together into one composite image. First, the transformation between two images was computed via the RANSAC method. Despite the somewhat small amount of overlap on the provided images, a relatively high-quality transformation was found. Then, the two images were projected onto a planar surface, after which blending techniques were applied to create a seamless stitch whilst aiming to minimize artefacts and visible distortions. Here it was briefly shown that the types of blends used - linear and sigmoidal blur - can yield different results depending on the source images. In order to improve the methods, we could either consider transformations with different parametrizations to achieve higher quality alignments between images, while more comprehensive blending techniques could be incorporated, such as Laplacian pyramids, and also additional post-processing such as gain compensation, to improve the quality of the final stitched image.

References

[1] Richard Szeliski. Image alignment and stitching: A tutorial, 2006.

[2] David G. Lowe. Object recognition from local scale-invariant features, 1999.

[3] Alexander Mordvintsev & Abid K. Feature matching — opencv-python tutorials.

[4] Martin A. Fischler & Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, 1981.