Panoramic Image Stitching
Computer Vision Final Project Report

Corentin Berteaux
CentraleSupélec
corentin.berteaux@student-cs.fr

Abstract

The panoramic image stitching problem consists in assembling multiple images of a same place into a larger panorama which offers a wider field of view. With two images, this problem only consists in finding the homography of one image relatively to the other. However, obtaining natural-looking panoramas require not only to work with more than 2 images, but also to add some compensation in order to reduce the differences between those images. In this paper, we implement a Python version of the AutoStitch algorithm [3]. Our implementation is able to create multiple panoramas from a single set of images, using SIFT and RANSAC, while doing gain compensation and blending in order to give a more natural look to the panorama. After showing the results on a few examples, we identify some of the problems that can still occur in specific cases, mainly because of a too large distortion or a blur in high frequency details.

The implementation can be found on GitHub at https://github.com/CorentinBrtx/image-stitching [1].

1. Introduction / Motivation

Panoramic image stitching consists in combining multiple images of a same location taken in different directions, into a single large image called a panorama. Basically, when considering a problem with only two images, the idea is to estimate a 3x3 homography (also called camera matrix) for one of the two images, while the other one is taken as the reference.

However, this problem gets more complicated when more images are involved. It such cases, an approximate placement of the images can be required from the user in order for the algorithm to produce a good enough panorama: for example, the images must be aligned along an axis, either horizontal or vertical. New methods can produce nice panoramas with few to none additional information from the user, even working with images arranged in any order.

Another important characteristic that must be considered when building such panoramas, more than the arrangement of the images itself, is to adapt the looks of each image so that there is no visible frontier in the result. Indeed, even if the images are taken at the same time, exposition or contrast can be slightly different, and other effects can come into play such as vignetting.

1.1. Use cases

The use cases for this problem are numerous. The main one is to let anyone take a beautiful panorama of a building or any location using its phone, which can allow capturing a larger field of view. Those panoramas can also offer 360° views of interiors or museums which can be useful for remote visits (especially relevant during Covid). Finally, some applications can also be found in the scientific domain, for example with images of space taken with telescopes, or images of the earth taken from space, that can be put together to offer a wider view of our universe.

2. Problem definition

First, one of the difficulties of this problem is that there is no real way to evaluate an image stitching algorithm. Unlike a problem of classification or regression, there is no metric (or at least no metric that is being widely used and accepted), and methods are often compared on specific examples that showcase the differences between the panoramas (distortion of the shapes, misalignment, etc.). Therefore, we proceed in the same way, by showing the effect of the algorithm on a few examples rather than by providing numbers.

That being said, the problem is rather simple: given two images or more, the model must be able to combine them into one single larger image, while making the transitions seamless. In our case, we also consider the possibility that the given images do not belong to the same panorama, thus resulting into multiple panoramas being given by our algorithm.

3. Related work

A lot of work has been done in image stitching over the years, proposing methods that are more robust and produce better results. While the first methods relied on user-provided information to produce good looking panoramas, a big step forward was taken in 2007 with the paper on AutoStitch by Matthew Brown and David G. Lowe [3], along with their proprietary software tool that could allow anyone to produce great panoramas simply by uploading their images in any order. There algorithm was based on the features detector and descriptor SIFT [7], created by Lowe in 2004, and the RANSAC algorithm [2].

Since then, multiple other methods have surfaced, each one proposing better results in edge cases by computing the stitching in a slightly different manner. Among the most popular recent methods, we can mention the APAP [8] and the AANAP [6] algorithms.

Even though the most recent methods presented above present better results, we propose in this paper to implement a python version of the AutoStitch algorithm. Indeed, there is not really any clean open-sourced implementation of this algorithm yet, and the other more recent methods are far too complicated to implement anyway considering the scope of this projected. This is especially true since AutoStitch already offers results that are more than acceptable.

4. Methodology

The AutoStitch algorithm [3] consists in multiple steps, and while we did not implement all of them as presented in the original paper, we focused on the ones that would improve the final panoramas the most. Thus, our implementation contains the following steps:

- Feature Matching using SIFT
- Image Matching with RANSAC
- Gain compensation
- Linear and Multi-Band Blending

4.1. Feature Matching

The first step is to characterize each image by extracting local features. To do so, we use the SIFT extractor [7], which for each image gives us a the locations of a certain number of keypoints, along with a descriptor for each of these keypoints. The main reason for using SIFT instead of other methods such as the Harris Corner Detector [4] is that SIFT features are invariant under rotation and scale changes, which means that our model can handle images with varying orientation and zoom.

Once features have been extracted from all images, a match is being calculated for each pair of images. To do so, each feature of one image is matched to its 2 nearest neighbors in feature space from the other image. Then, the Lowe's ratio test is used to keep only relevant matching features. This test consists in comparing the first nearest neighbors to the second, and keeping only the matches where the distance with the closest neighbor is significantly smaller than the distance with the second closest one (significantly being determined by the value of the scaling factor used). This test does sort of a soft thresholding that eliminates most of the false matches.

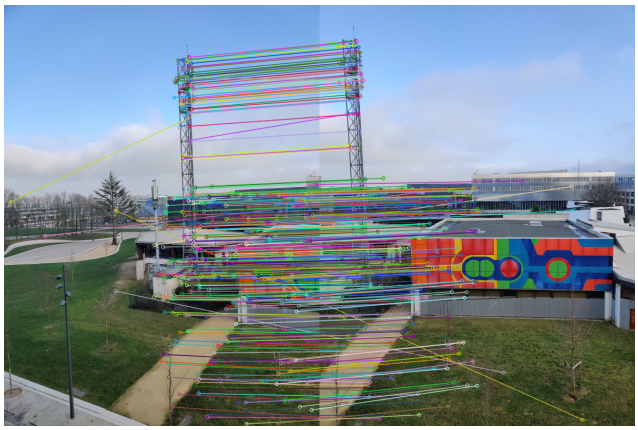


Figure 1: SIFT Features matches between two images

After this step, we have for each pair of images a list of matches, each one composed of a feature from each images (see Fig 1). The next step is to use those matches to identify the images that would overlap each other in the panorama, and to compute homography matrices.

4.2. Image matching

4.2.1 Robust Homography Estimation using RANSAC

RANSAC (random sample consensus) [2] is a robust estimation procedure that uses a minimal set of randomly sampled correspondences to estimate image transformation parameters, and finds a solution that has the best consensus with the data. It works with multiple iterations. At each iteration, a set of 4 feature matches is randomly selected, and is used to compute the homography **H** between them using the direct linear transformation method [5].

This operation is repeated a certain number of times, each with a different randomly selected set of 4 samples. Then, the homography with the highest number of inliers (*i.e.* matches whose projections are consistent with H within a tolerance ϵ pixels) is selected. It can be shown that, for a pair of images where half of the matches are corrects, the probability to have **not** found the correct homography after 500 iterations is approximately 1×10^{-14}, thus making it a pretty robust method.

4.2.2 Probabilistic Model for Image Match Verification

At this stage, each pair of images consists of a list of features matches and an homography for one image relatively to the other. However, we don't know yet which pairs of images should indeed overlap in the final panorama, and which pairs actually don't have any area in common. To verify the image matches, we compare the features matches that are consistent with the homography (RANSAC inliers) with the features matches inside the area of overlap between the two images but not consistent (RANSAC outliers).

For a given image we denote the total number of features in the area of overlap n_f and the number of inliers n_i. Then, by making small assumptions on the distributions of the feature matches, we can estimate the probability that the image match is correct using these two numbers, and obtain a condition for an image match to be correct:

$$n_i > \alpha + \beta * n_f \tag{1}$$

The detailed calculus can be found in the AutoStitch original paper [3]. The parameters α and β could theoretically been adjusted based on the data, but we used the provided values, that is $\alpha = 8.0$ and $\beta = 0.3$.

Using the above condition, we can keep only the correct image matches, and divide the data into connected components (*i.e.* sets of images that are linked together by matches). This allows us to recognise multiple panoramas in a set of images, and reject noise images which match to no other images (see Fig 2).

4.3. Gain compensation

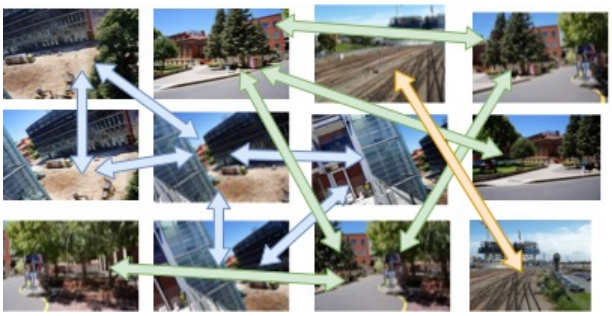
The two previous steps already gives us a panorama, where each image is placed correctly relatively to each other. However, the result images are far from being good-looking, and the transitions between each image is nothing but seamless. To reduce the differences, we compute the overall gain between images, and apply compensation to reduce them.

To do so, for each panorama, we consider a global error function defined over all the images using their areas of overlap as follow:

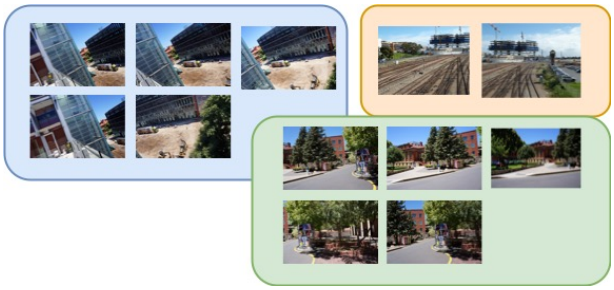
$$e = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} N_{ij}\left(\frac{(g_i\bar{I}_{ij} - g_j\bar{I}_{ji})^2}{\sigma_N^2} + \frac{(1-g_i)^2}{\sigma_g^2}\right) \tag{2}$$

where $N_{ij} = |\mathcal{R}(i,j)|$ is the number of pixels in image i that overlap in image j, $\bar{I}_{ij}$ is the mean intensity of the image i in the overlap region with image j, defined as

$$\bar{I}_{ij} = \frac{\sum_{u_i \in \mathcal{R}(i,j)} I_i(u_i}{N_{ij}} \tag{3}$$



(a) Valid images matches are identified



(b) Connected components are computed

(c) The three final panoramas

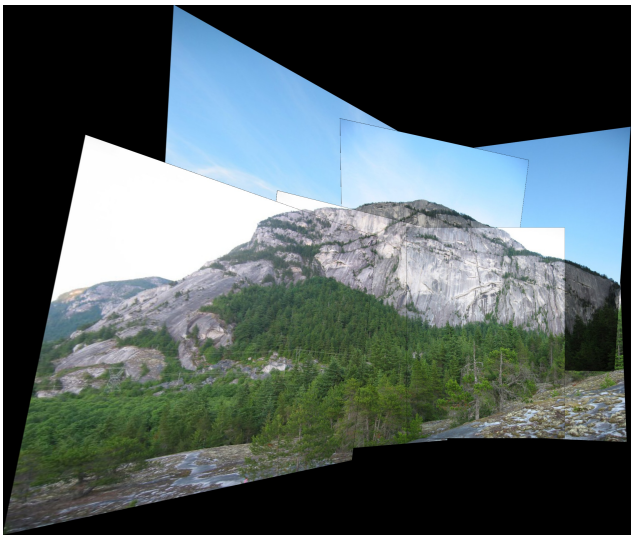
Figure 2: Image matching process

and g_i represents the gain for image i (this is what we want to compute). The parameters σ_N and σ_g are the standard deviations of the normalised intensity error and gain respectively. We use the values given in the paper, $\sigma_N = 10.0$, $(I \in \{0..255\})$ and $\sigma_g = 0.1$.

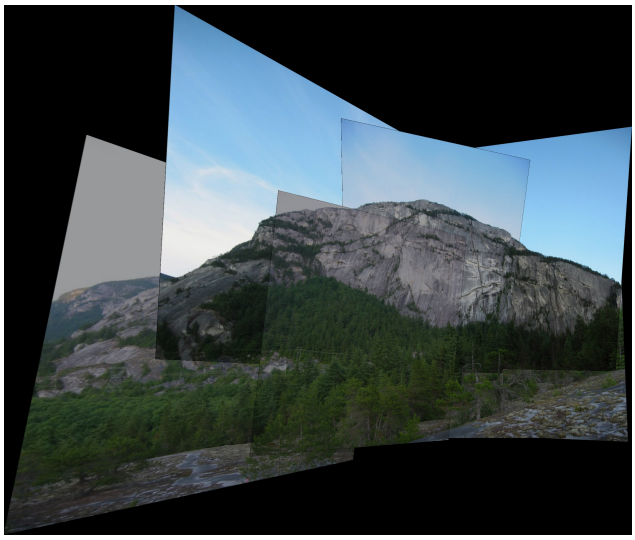
To solve this quadratic function, we set the derivative to 0, which gives:

$$g_j\left(\frac{2}{\sigma_N^2}\sum_{i=1}^{n} N_{ji}\bar{I}_{ji}^2 + \frac{\sum_{i=1}^{n} N_{ki}}{\sigma_g^2}\right) - \frac{2}{\sigma_N^2}\sum_{i=1}^{n} N_{ki}\bar{I}_{ki}\bar{I}_{ik}g_i - \frac{\sum_{i=1}^{n} N_{ki}}{\sigma_g^2} = 0 \tag{4}$$

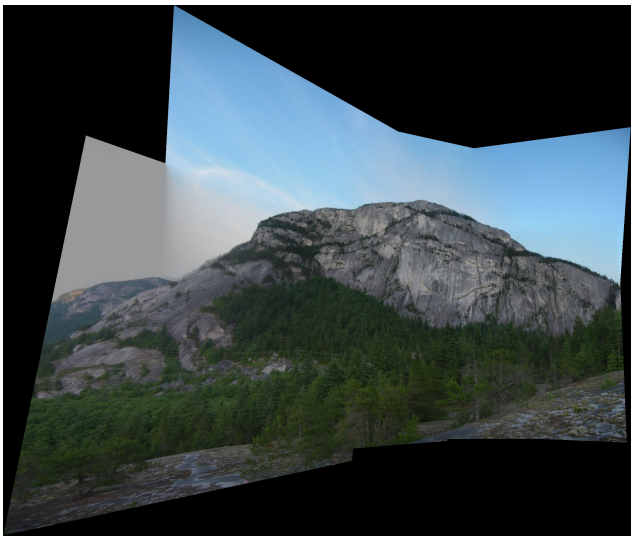
that can eventually be solved with a simple linear solver (see Fig 3b).



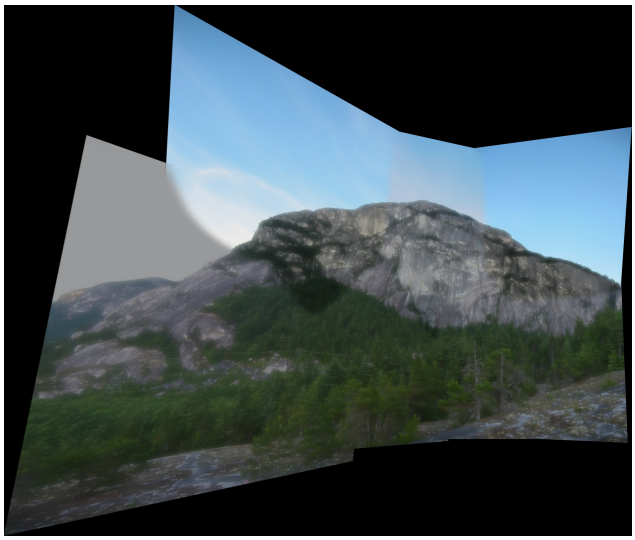
(a) No compensation



(b) Gain compensation



(c) Gain compensation and linear blending



(d) Gain compensation and multi-band blending

Figure 3: Different steps to reach the final look of the panorama

4.4. Blending

4.4.1 Linear Blending

While the gain compensation presented in the last section allows to have already better results, the transitions between the different original images is still visible in most panoramas. This can be due to many other effects (misalignment, vignetting, etc.). To further reduce these artefacts, we use a simple blending strategy that uses weights matrices to give more importance to images as we are close to their center, and less importance on the edges. This allows to do a better stitching than simply putting them on top of each other.

For each image, we consider a weight matrix $W_{ij} = w_i w_j$ where w is a vector that is equal to 0 on both ends and to 1 at the middle. Then, for each pixel of the panorama (x, y), we compute its value as a weighted sum of all the other images (considering that the weight matrices of each image outside of their actual location is equal to zero):

$$I(x,y) = \frac{\sum_{i=1}^{n} I_i(x,y) W_i(x,y)}{\sum_{i=1}^{n} W_i(x,y)} \tag{5}$$

This blending method allows to smooth the transitions between each image and to make the panorama much more natural (see Fig 3c). However, the weighted sum means

that multiple images contribute to the same pixels, and small misalignments can sometimes mean that the same detail will be shown in two different places by two different pictures, thus leading to blurring of high frequency detail. To solve this problem, AutoStitch proposes the Multi-Band blending strategy.

(a) Panorama details with linear blending

(b) Panorama details with multi-band blending

Figure 4: With linear blending, details come from two images and result in the details being shown in twice. With multi-band blending, the details are not shown twice, but the image is overall more blurry and overexposed. The transition between the two images is also more abrupt.

4.5. Multi-Band Blending

The idea behind multi-band blending is to blend low frequencies over a large spatial range, and high frequencies over a short range. This way, the small details will be the responsibility of a single image, while the blending of low frequencies still allow to smooth the transition between images.

First, we initialise blending weights for each image by finding the set of points for which image i is most responsible:

$$W^i_{max}(x,y) = \begin{cases} 1 & \text{if } W^i(x,y) = \max_j W^j(x,y) \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

These max-weight maps are successively blurred to form the blending weights for each band, along with a similar blurring performed on the image itself to build each band. For the first band, a high pass version of the rendered image is formed:

$$I^i_\sigma(x,y) = I^i(x,y) * g_\sigma(x,y) \quad (7)$$
$$B^i_\sigma(x,y) = I^i(x,y) - I^i_\sigma(x,y) \quad (8)$$

where g_σ is a Gaussian of standard deviation σ, and $*$ denotes convolution. In doing so, B_σ represents spacial frequencies in the range of wavelengths $\lambda \in [0, \sigma]$. The weight matrix associated with this band is computed using the same convolution:

$$W^i_\sigma(x,y) = W^i_{\max}(x,y) * g_\sigma(x,y) \quad (9)$$

Then, other bands are computed iteratively, using a standard deviation $\sigma' = \sqrt{(2k+1)}\sigma$ that is getting larger, with $k \geq 1$:

$$I^i_{(k+1)\sigma} = I^i_{k\sigma} * g_{\sigma'} \quad (10)$$
$$B^i_{(k+1)\sigma} = I^i_{k\sigma} - I^i_{(k+1)\sigma} \quad (11)$$
$$W^i_{(k+1)\sigma} = W^i_{k\sigma} * g_{\sigma'} \quad (12)$$

This way, each band k represents spacial frequencies in the range of wavelengths $\lambda \in [\sqrt{(2k-1)}\sigma, \sqrt{(2k+1)}\sigma]$. The bands for the overall panorama are obtained by linear combination of the different images:

$$I^{multi}_{k\sigma}(x,y) = \frac{\sum_{i=1} nB^i_{k\sigma}(x,y)W^i_{k\sigma}(x,y)}{\sum_{i=1}^n >^i_{k\sigma}(x,y)} \quad (13)$$

For the final band, we simply take what is left of the image in order to represent all the low frequencies in the image:

$$B^i_{K\sigma} = I_{(K-1)\sigma} \tag{14}$$
$$W^i_{K\sigma} = W_{(K-1)\sigma} \tag{15}$$

Finally, we add the different bands together to obtain the final panorama (see Fig 3d).

More details and illustrations about this method can be found in the AutoStitch paper [3].

5. Evaluation

Even though there is no real metric when it comes to image stitching, as explained before, the implemented algorithm can be used on pretty much any type of images, as long as the total field of view is not too large (which would result in distortion, but more on that later). In most cases, the results with a simple blending are very satisfying, and the demarcations between images are almost always seamless, provided there is no big change in the environment (see Fig 6). However, we can observe a few problems that could be resolved by implementing more complex methods.

5.1. Problem of distortion

In our image stitching algorithm, we arbitrarily choose one image as the image of reference, and all the other images' homographies are then computed with regard to this image. Even though some heuristics were implemented to choose a "good" image of reference (we choose an image with a lot of matches, which means there is a good chance it is at the center of the panorama), there are still cases where this choice is not good enough. This leads to a distortion too important, which can make an unpleasant result (see Fig 5), and sometimes even make the size of the panorama tends to infinite. Solving this problem would require either choosing a better image of reference, or making transformations to the panorama in order to display it in a better way.

5.2. High frequency blur

An other problem, which was mentioned before, is the blurring that can occur in the details of the panorama. This problem is supposed to be solved with multi-band blending instead of linear blending. However, even though some of this blur indeed disappear when using multi-band blending, the overall aspect of the result is not as good, with an image that tends to be over-exposed, blurry, and the transitions between images is sometimes a bit messy. A better fine-tuning of the parameters used during blending (*i.e.* the number of bands and the standard deviation) may help to improve the results on this aspect.

6. Conclusions

This project was the opportunity to implement an image stitching algorithm from scratch, using already implemented methods only for very specific operations that would have taken too much time to implement. The final algorithm implemented in Python is very efficient, with a lot of advanced features that lead to a better experience:

Figure 5: Distortion in a panorama

- Image matching with SIFT and RANSAC allows to recognize multiple different panoramas from a set of images, and can also eliminate any noisy image that is not included in any panorama
- Gain compensation gives a more natural look to the final panorama, reducing the differences between images
- Linear and multi-band blending further improve the look of the panorama by making the transitions between images smoother, even though the results obtained with multi-band blending are a bit disappointing.

6.1. Future work

More advanced methods have been proposed since AutoStitch, focusing on the problems that were identified in the Evaluation section above. For example, instead of simply computing homographies from image to image, a mesh grid is used to compute a more general arrangement of the images, and the reduce any error and improve the overall look of the panorama. A nice way to continue this project would be to see how these new methods could be integrated into our implementation, and how they would indeed increase the quality of the results.

Figure 6: Different panoramas obtained with linear blending

References

[1] Corentin Berteaux. Image stitching, 2022. https://github.com/CorentinBrtx/image-stitching. 1

[2] Robert C Bolles and Martin A Fischler. A ransac-based ap-

proach to model fitting and its application to finding cylinders in range data. In *IJCAI*, volume 1981, pages 637–643. Citeseer, 1981. 2

[3] Matthew Brown and David G Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, 2007. 1, 2, 3, 6

[4] Konstantinos G Derpanis. The harris corner detector. *York University*, 2, 2004. 2

[5] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 2

[6] Chung-Ching Lin, Sharathchandra U Pankanti, Karthikeyan Natesan Ramamurthy, and Aleksandr Y Aravkin. Adaptive as-natural-as-possible image stitching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1155–1163, 2015. 2

[7] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 2

[8] Julio Zaragoza, Tat-Jun Chin, Michael S Brown, and David Suter. As-projective-as-possible image stitching with moving dlt. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2339–2346, 2013. 2