Feature Detection in Image Stitching

MATTHEW HALEN CRANKSHAW

Institute of Information & Mathematical Sciences
Massey University at Albany,
Auckland, New Zealand
14303742@massey.ac.nz

* *Encyclopedia Galactica, Research Division*
Betelgeuse, Ursa Minor
T.Author@encgal.edu

Feature detection is the most critical part of Image Stitching, an effective feature detector needs to find the most valuable features as quickly as possible whilst disregarding unimportant features such that the image stitch can be seamless.

Keywords: Computer Vision; Feature Detection; Image Stitching; ORB, SURF, SIFT.

1 Introduction

This mini-project will be exploring the different methods of Feature detection and feature matching in the context of Image stitching. Image stitching is an important technique that is used very often in the modern day. Most of the challenges that face Image stitching is incorrect the feature detection stage or the feature matching stage, these stages have a large effect on the quality of the features themselves. In this report I will be talking about different feature detection algorithms as well as each one's strength and weakness. As to provide better understanding on the use of these algorithms and their place in image stitching.

2 The Image Stitching Process

2.1 Feature Detection

Feature detection is the step that I will be focusing the most on in this project as it is the most dependent part of the process that really defines a good image stitch or not. The process involves finding key points in each image which are robust and can be found and matched in both images. I will be talking about this step in more detail, specifically about three different feature detection algorithms. SIFT, SURF and ORB. I will be talking about how these algorithms differ and the results they are able to achieve. Depending on the feature detection the resulting panoramas quality should increase or decrease. Below is an image showing features that have been detected on a greyscale image for efficiency.

Figure 1: Greyscale image showing detected features

2.2 Feature Matching

Feature matching is the process of taking many features from each image that is going to be stitched together and matching these features such that you can be sure that these two features are in fact relating to the same "space" in the image. I will be testing two different types of feature matching algorithms as to find which one is the most accurate and the fastest. The two feature matchers I will using are the brute-force matcher and the FLANN (Fast Library for Approximate Nearest Neighbours) based matcher. Below you can see an example of features being matched between two images.

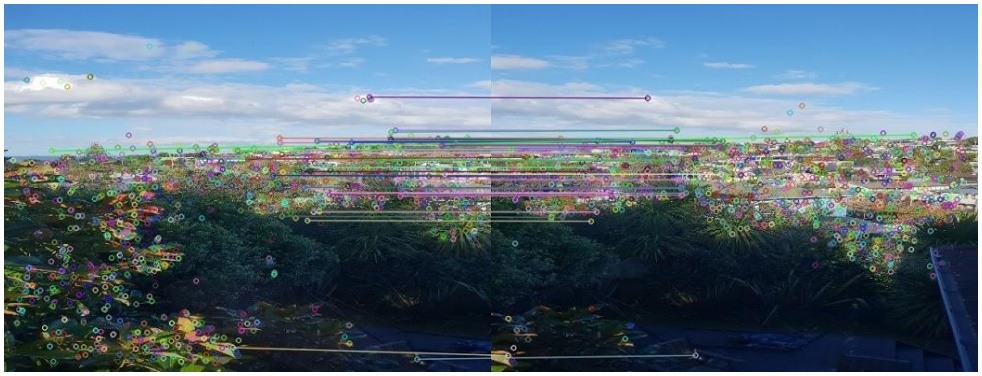

Figure 2: Image Showing Features Detected in Two Images and their Matches

2.3 Finding Homography

Any two images are said to be related by a homography if the two images are of the same planar surface in space, it can also be said that they are related by a homography if they share an overlapping field of view. For instance, in the image in Figure 2 these two images are related by a homography because they share an overlapping field of view.

2.3.1 The Math

If we have two cameras: *a* and *b*

Looking at point: P_i

From the projection of P_i in *b*: $${}^{b}p_i = ({}^{b}u_i; {}^{b}v_i; 1)$$

To the projection of P_i in *a*: $${}^{a}p_i = ({}^{a}u_i; {}^{a}v_i; 1)$$

$$ {}^{a}p_i = \frac{{}^{b}z_i}{{}^{a}z_i} K_a \cdot H_{ba} \cdot K_b^{-1} \cdot {}^{b}p_i $$

where ${}^{b}z_i$ and ${}^{a}z_i$ are the z coordinates of P in each frame and where the homography matrix H_{ba} is:

$$ H_{ba} = R - \frac{tn^T}{d}. $$

Homography assumes a pinhole camera model which not the case in modern cameras however it is a close enough approximation that we can get accurate results.

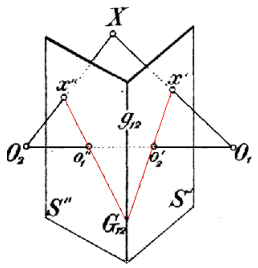

Figure 3: Figure Showing the Homography of two Planes

2.4 Perspective Transform

One of the final steps in image stitching is to warp the perspective of the images in such a way that they can be overlaid on top of each other and they will fit almost seamlessly this is done by using the homography matrix of each image to warp its perspective in 3D space and then fit them together in the same image this is said to be the "stitching" part of the process. One of the challenges that comes up when "stitching" these images together is that they there can be an overlapping due to different exposures which I will be talking about later.

2.5 Cropping and Filtering

Cropping is not a necessary step in the image stitching process however it is important especially when there are multiple images stitched together. The reason cropping is necessary is that practically when you are taking a panorama of any kind you are taking multiple images of a 3-dimensional world and turning it into a 2-dimensional image. The result when these images are stitched together is one image with lots of empty space as seen below, although this gives a good impression of the 3D composition of the image it does not look good.

Filtering is also not a required step however it greatly improves the quality of the resulting panorama. When images are stitched together in a panorama and the lighting and exposure changes with each different image you can get a layering effect. I will be talking about this in more detail in the Challenges and Issues section.

The reason this filtering step is important is because we do not want to take away any of the vividness or detail in the image we just want to synthetically even the exposure and contrast of each image in such a way that they seem like the same image.

3 Feature Detection Methods

In this section I will be briefly be discussing how each feature detection method works and theoretically how they should behave under different conditions. In the Testing and Results section I will be discussing how the methods actually performed in the real world taking note of both their speed and accuracy.

3.1 SIFT (Scale-Invariant Feature Transform)

As per the name SIFT strives to be a Scale Invariant, Rotation Invariant, semi Illumination Invariant and viewpoint invariant. SIFT is quite an in-depth algorithm it strives to cover all of the bases and yield very accurate and robust features. The process goes as follows:

- Construct a scale space
- Laplacian of Gaussian approximation
- Find key points
- Get rid of bad key points
- Assign orientation to the key points
- Generate SIFT features

Advantages:

SIFT's features are very robust, because of the amount of invariance that SIFT has it is able to produce features that are match able in very different images.

Disadvantages:

SIFT is said to be slower than the other types of feature detection, It does a lot more work than the others so it will take more time. However, what I am testing for is whether getting less more accurate (SIFT) features is better than getting more less accurate ones, for instance with the other algorithms.

3.2 SURF (Speeded-up Robust Features)

SURF is like SIFT and is sometimes said to be and "Approximation" of SIFT. SURF goes through the same steps and SIFT however has some differences. The algorithm has three main parts:

- Interest point detection
- Local Neighbourhood description
- Matching

3.3 ORB (Orient FAST and Rotated BRIEF)

ORB is an open source algorithm developed for the OpenCV library unlike SIFT and SURF which have been patented.

ORB is said to be a fusion of FAST key point detector and the BRIEF descriptor hence the name, although it has a lot of modifications. One of the issues with the FAST key point detector is that it is not rotation invariant, this is where ORB has modified it in such a way that it is rotation invariant.

The resulting algorithm is very fast however because of it being based on FAST and BRIEF it is not as robust against scale and rotation variance.

4 Feature Matching Methods

In this section I will briefly explain the difference between each of the two feature matching methods in the Testing and Results section I will discuss how they compare in speed as well as accuracy. I will also decide which one would be best suited for carrying out the testing of the feature detection and image stitching for this project. The goal of the matching algorithms is to find the nearest neighbour of the points in the other image.

4.1 Brute-Force Matcher

The brute-force matching is the simpler of the two matching algorithms. The brute force matcher takes the point of interest in one image and matches it against all of the points in the other image, it then does some distance calculation to find out which is the closest and uses that one. Although this is a good way of finding the closest point it can take a very long time to match large data sets.

4.2 FLANN Based Matcher

FLANN is an abbreviation for Fast Library for Approximate Nearest Neighbours. It contains a collection of different algorithms for finding nearest neighbours. Of which there is a machine learning KNN (K Nearest Neighbours) algorithm which is faster than that of a Brute-Force Matcher especially when the dataset is large.

5 Challenges and Issues

Figure 4: Image Showing Motion tearing, Incorrect Feature Detection and Difference of Exposure.

5.1 Incorrect Feature Detection/Matching

Incorrect Feature Detection is the part that I will be paying the most attention to in this mini-project. This occurs when a key point or feature is detected that is either not a good point of interest or is incorrectly matched with another completely different key point/feature. This results in an incorrect or inaccurate warp transform causing a visual tear in the image as seen in Figure 4 above.

5.2 Movement Tearing

Movement tearing is where you have two images with an overlapping field of view however there is some object in each of the images which has moved between the time the image was taken in the first image and the second image. This problem can be overcome in two ways. The first is kind of obvious, make sure that nothing is moving in the scene, the second way which can be used if the first way cannot be avoided, is the take both images simultaneously this may require two separate cameras. An example of image tearing can be seen in Figure 4 above.

5.3 Difference of Exposure

Difference of exposure is a simple problem but can prove to be quite tricky to solve. The problem occurs when two different images are taken with two different exposures and then when the images are stitched together they have a slightly different exposure which can be very noticeable and certainly not pleasing to the eye. There are a few different ways to solve these two effective ways seem to be alpha blending and Poisson blending. An extreme case of Difference of Exposure can be seen in figure 4 above.

6. Testing and Results

6.1 Feature Matching

For the below graphs I have run a simple loop to time how long it takes the feature matching algorithms to match the features using increasingly larger amounts of features. In the first graph I have 50 to 30000 features, in the second graph I have 20 to 1000 features. I have chosen these numbers to represent the strengths of the different algorithms. Looking at Figure 5 below, it would seem as though there is no competition between Brute-force and FLANN, FLANN clearly wins by considerable amount, at 30000 features it takes one third the time that it takes Brute force, it is critical to know this, however it is important to consider that these numbers are somewhat unrealistic for image stitching. We will not be needing 30000 features; therefore, I have made another plot shown in Figure 6, with a more realistic comparison.

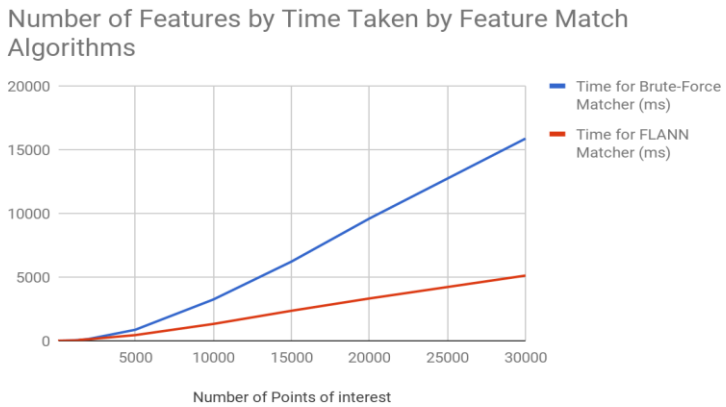

Figure 5: Showing Brute Force and FLANN feature matching over time for up to 30000 features

Above we can see below in Figure 6, there are only 20 – 1000 features, this is more in the ballpark of what amounts of features we would be dealing with. As you can see quite clearly brute-force has quite an exponential curve and takes considerably less time than FLANN until around 750 features, by which time the advantages of FLANN start to become apparent. It is also important to note that while Brute force is quite a sharp and very stable rise, FLANN on the other hand is very jagged and random, this could pose a problem if you are needing a very consistent compute time and are dealing with different numbers of features.

So, which one is better for the purposes of this project. Well for this project I have decided to go with FLANN, FLANN seems to excel in the higher number of features area, and during this project I will be testing those amounts of features so I will need a matching algorithm that will do those relatively quickly, and the other reason is that Brute force only beats FLANN by about 5-8ms in the range of 10 - 600 features which is not a big difference and further validates my decision.

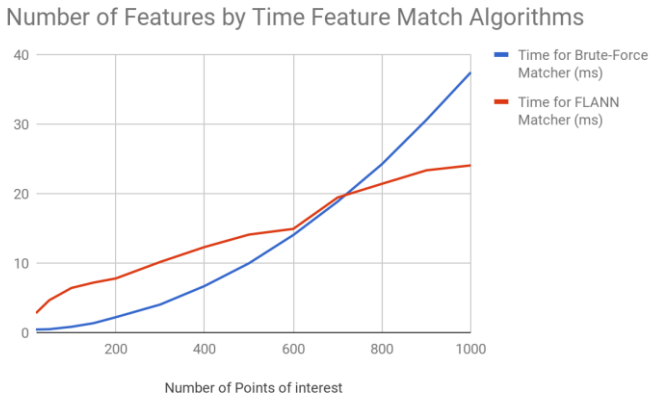

Figure 6: Showing Brute-Force and FLANN matchers over time from 20 to 1000 features

6.2 Feature Detection

Above in the Feature detection Methods section I have already discussed the differences between the types of algorithms, so in this section I will be covering the differences in performance with regards to speed and number of features found for SIFT, SURF and ORB.

As there are many parameters and each algorithm can be tweaked slightly through these parameters I will be using the default parameters as specified by OpenCV 3. The parameters I will be tweaking myself are the number of features parameter in both SIFT and ORB and the hessian parameter in the SURF.

I will be testing the speed as well as the number of features the algorithm is able to generate. And conclude with which feature detection algorithm would be best suited for image stitching.

Straight off the bat when it comes to testing we have a problem, SIFT and ORB allow you to provide a “number of features” parameter whereas SURF asks for a “min hessian”, this makes testing slightly more complicated, a question being, how does one test the speed if you cannot pick exactly how many features you want. So, I will be testing SURF slightly differently.

6.2.1 Number of Features Found

As you can see in Figure 7 below, I have tested the number of features provided when calling ORB, SIFT and SURF. For ORB and SIFT I have given 1000 as the number of features to find. Ideally, we would like to see a consistent number of features close to that amount for it to be useful in image stitching. As you can see SIFT is very good at providing a consistent number of features for most images giving exactly 1000 features, in fact there is only one case where SIFT failed to deliver enough features.

With ORB the numbers are also consistent, however It seems the ORB algorithm tends to provide more features than necessary. Which is not a bad thing. On the other hand, SURF is quite bad at this, the number of features is very sporadic and sometimes fails to provide any features and other times provides over 1000 features. Even with some tweaking I was unable to get SURF to provide a consistent number of features. In saying this, if you are after many features, SURF can pump out thousands of features if the min hessian parameter is tweaked.

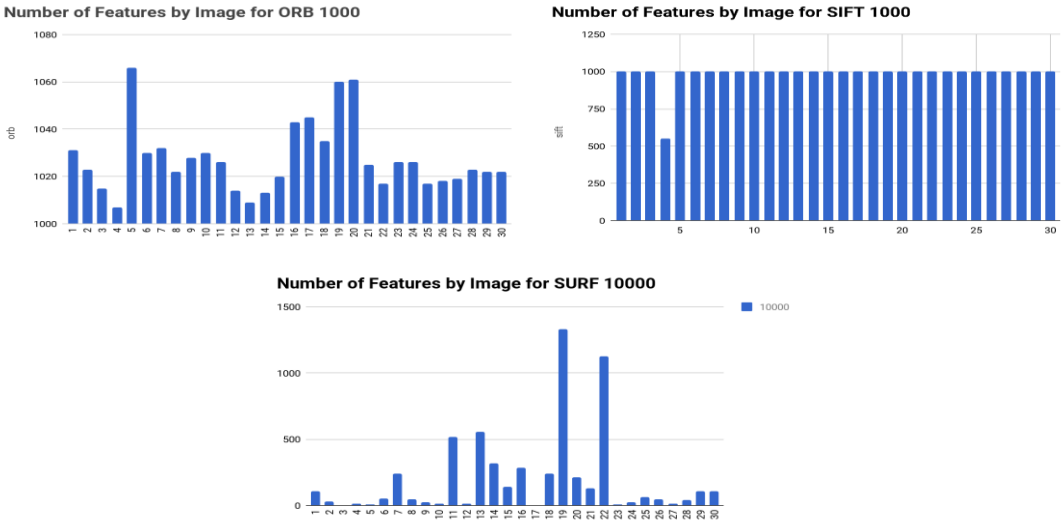

Figure 7: Graphs Showing the Number of Features Found by SIFT, SURF and ORB for each image.

6.2.2 Time Taken to Find Features

Looking at Figure 8, when it comes to the amount of time to find features the results surprised me, ORB can provide a more than adequate number of features and performs very well taking almost one tenth the speed of SIFT, surprisingly this does not seem to change when increasing or decreasing the number of features.

SIFT performed as expected, it took quite a bit longer than ORB but when looking at the previous graph you can see that it provided the most consistent number of features. With further testing I also noticed that SIFT keeps very a consistent time even for a higher number of features always keeping around to two to four second mark.

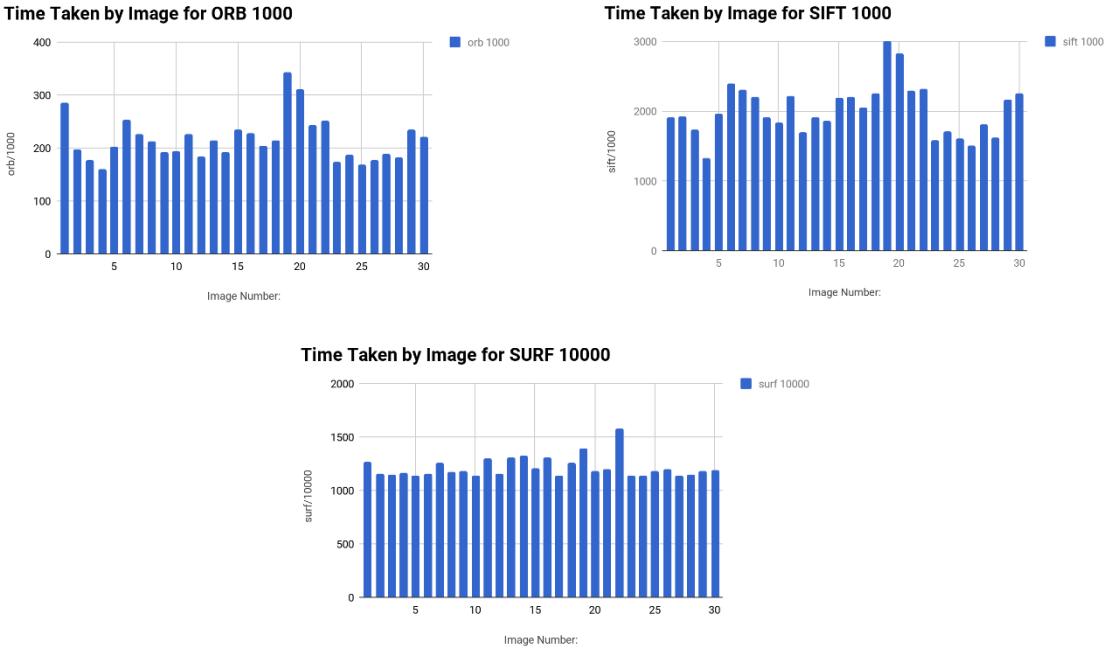

Figure 8: Graphs Showing the Time Taken for ORB, SIFT and SURF for each Image

6.2.3 In Practice

The theoretical speed and number of features is an important aspect of a good image stitch but in practice the feature detecting algorithm needs to deliver. In this section I will talk about each algorithm with each image stitch. I have got 30 pairs of images which I will testing and I will give a score of either 1 or 0, 1 meaning the resulting stitch is good and there are no visible tears in the image.
I will then also give the amount of time it took to achieve these results.

It is important to note that the following times are not considering the time it takes to feature match or to find the homography. These processes will take more time as the number of features go up. Ideally, we would like a feature detector which returns a small number of features that are accurate and relatively quickly.

In the table below the number next to the detector name is the number of features or in the case of SURF, the min hessian.

Table 1: Scores and Time Taken for each Feature Detector

Detector	ORB 5000	ORB 10000	SIFT 500	SIFT 5000	SURF 1500	SURF 500
Score	18/30	22/30	22/30	27/30	22/30	26/30
Average time taken Feature detection:	40.783 ms	43.350 ms	501.800 ms	561.8 ms	425.01 ms	764.82 ms
Average time taken for image stitch	230.06 ms	642.82 ms	564.96 ms	934.08 ms	694.09 ms	1897.94 ms

As you can see in the above table ORB is by far the fastest at detecting features however this comes at a price. It can detect thousands of features in no time at all however it doesn't quite seem to always get the robust ones.

SIFT on the other hand takes considerably more time but is able to detect very robust features even with one tenth as many as ORB.

SURF is a bit of a hard case. Although it is able to score well and seemingly on paper looks like a contender with SIFT it is very unstable, sometimes giving 500 features and other times giving over 20000. This kind of instability would be a big problem when it comes to applications which rely on a very consistent flow of features.

3 Conclusion

In conclusion, each feature detector has their own strengths and weaknesses, and all of them would have their own application somewhere.

I do believe that SIFT with its consistent, robust and predictable feature detection, is likely the best in most situations. It has topped the scores and is able to get 5000 features in under a second and get an accuracy of 27 out of 30.

ORB I would say is incredibly fast, its purpose would lie in applications where you need a fast stream of thousands of features, for instance, real time applications. ORB will perform; however it does lack in accuracy, and for image stitching the accuracy is not enough to give consistent results, at least not without some further tweaking.

As for SURF, this seems to be the black sheep of the bunch, it is incredibly accurate rivalling that of SIFT, however, it is volatile in nature; this would be a great algorithm for applications where “no features” would be a possible and likely outcome, and you would need it to scale appropriately as the scene becomes more complex, however for image stitching, it is far to inconsistent and “no features” is not a possibility nor an acceptable outcome.

References

[1] Rosebrock, A. (2018). OpenCV panorama stitching - PyImageSearch. [online] PyImageSearch. Available at: https://www.pyimagesearch.com/2016/01/11/opencv-panorama-stitching/ [Accessed 22 Jun. 2018].

[2] En.wikipedia.org. (2018). Scale-invariant feature transform. [online] Available at: https://en.wikipedia.org/wiki/Scale-invariant_feature_transform [Accessed 22 Jun. 2018].

[3] En.wikipedia.org. (2018). Speeded up robust features. [online] Available at: https://en.wikipedia.org/wiki/Speeded_up_robust_features [Accessed 22 Jun. 2018].

[4] Sinha, U. (2018). Implementing SIFT in OpenCV - AI Shack. [online] Aishack.in. Available at: http://www.aishack.in/tutorials/implementing-sift-opencv/ [Accessed 22 Jun. 2018].

[5] Opencv.org. (2018). OpenCV library. [online] Available at: https://opencv.org/ [Accessed 22 Jun. 2018].

Source Code: https://github.com/Accusmus/mini-project.git