Video stabilization for UAV land surveying

Isawan Millican
ppyim@nottingham.ac.uk

Christopher Cole
ppycjco@nottingham.ac.yk

November 24, 2017

Abstract

The recent growth of drone technology has sparked an interest in the video stablisation technologies. We present an algorithm based on ORB feature point matching to perform video stablisation. The algorithm is demonstrated to generate panoramic map images using drone video footage. An error analysis of our developed method is given and we explore some of the visual artifacts that can occur as a result of our stablisation algorithm.

1 Introduction

1.1 Background

In recent years, unmanned aerial vehicles (UAVs), commonly referred to as drones, have become popular amongst consumers. There has also been an upsurge in usage by emergency services. These drones are fitted with a camera to record flights, stream video footage, or to help with piloting. Unfortunately, cameras are subject to turbulence due to aerodynamic effects which create unwanted jitters in any videos. In more advanced drones, video quality is increased using a gimbal which attempts real time compensation of the turbulence, however a digital approach to video stabilisation provides useful information such as the motion model of the camera.

Video stabilisation can be used to compensate all motions or remove high frequency jitters. Removal of high frequency jitters is useful for applications in stabilization of video streams where the camera motion is unimportant such as astronomy where motion is caused by atmospheric effects. Our work is concerned with the former case, which has the advantage of providing contextual information on the surroundings of each frame in the video.

In this paper, we describe an algorithm to estimate the motion of the camera. An analysis of the numerical errors of the algorithm is performed. We demonstrate how the global motion information can be used to construct a panoramic image from drone footage. This method produces an image or video with significantly higher resolution than the original video dimensions. Applications of this include mapping of areas with greater resolution than is available with the camera being used, by recording a video of the drone passing over the area of the desired image.

Another possible use is to show the path taken by an object being tracked by the drone.

2 Method

2.1 Camera model

A pinhole camera model has been used. We assume the surface being imaged is sufficiently far away from the camera. This allows us to approximate any frame t as an affine transformation of the previous frame $t-1$. The model was restricted to a subset of affine transformations: a counterclockwise rotation about $(0,0)$ by angle θ, a scaling by factor s and spatial translations Δx and Δy. This improves the numerical stablity of the model by reducing the degree of freedom of the model to four. The transformation can be expressed by a matrix product when the screen coordinates (x, y) is augmented to $(x, y, 1)$.

$$\begin{bmatrix} x_t \\ y_t \\ 1 \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & \Delta x \\ s\sin\theta & s\cos\theta & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ 1 \end{bmatrix} \tag{1}$$

2.2 Transformation estimation

Transformation estimation involves finding an affine matrix that maps the coordinate of a frame t to the coordinate space of the first frame. Our algorithm is based on the feature point matching discussed by Kulkarni et. al. [1]. An overview of the developed algorithm can be described by the following.

for all Frames **do**
  Convert frame to greyscale
  Find frame's feature points and associated visual descriptor with ORB
end for
for all Consecutive frames **do**
  Match the keypoints from previous frame
  Estimate the interframe transformation matrix M_t with RANSAC
  Find the global transformation by $\prod_{i=0}^{t} [M_i]^{-1}$
end for

For each frame we identify the feature point and descriptor with ORB. ORB is a patent-free feature detector and descriptor based on FAST and BRIEF[2]. ORB detects corner features within the image with FAST, Harris corner detection is then used to reduce the number of features removing false positives. BRIEF is then used to assign a binary visual descriptor vector to each keypoints. Previous work shows comparable performance between ORB, SIFT and SURF[3]. Similarity between descriptors is given by the Hamming distance i.e. the number of positions in the vectors where corresponding symbols are different. We set the parameters such that a maximum of 500 features are extracted per frame.

We initially matched the keypoints in consecutive frames. The Hamming distance is calculated between the binary descriptors. The two nearest neighbours

of each point is found by a brute force algorithm. The ratio test proposed by D. Lowe[4] is applied to reduce false matches: if the Hamming distance ratio of the two nearest neighbours is greater than a 0.8 then reject both matchings to that point. Of the remaining matchings, we find the spatial position vectors from the frame $t-1$ keypoints to the frame t keypoints.

The remaining matches still contains many false results. The interframe transformation matrix is estimated using the RANSAC method[5]. RANSAC is an iterative method to estimate a best fit model of data with significant outliers. The estimation works provided that at least half of the matches are true matches. The result of this is a matrix M_t that transforms the previous frame $t-1$ into the frame t.

A global transformation must be calculated. The global transformation describes the coordinate mapping of frame t to the first frame. This is represented by an affine matrix M_t^G. The global transformation matrix is related to the interframe transformations by

$$M_t^G = \prod_{i=0}^{t} [M_i]^{-1} \tag{2}$$

A performance improvement is gained by writing the recursive definition

$$M_t^G = M_{t-1}^G [M_t]^{-1} \tag{3}$$

where we set M_0^G as the identity matrix and M_{t-1}^G the global transformation of the previous frame.

Finally, we perform a coordinate transform to map each frame to the initial coordinates. This is computed with a backward linear interpolation scheme.

2.3 Visualisation

To visualise the stablisation, we developed two visualisation algorithms. Also, a simple stablized algorithm was implemented.

2.3.1 Inplace frame stiching

The purpose of this algorithm is to create a single video which is a stitch of all video frames. As the video plays, the current frame is inserted last. This would provide the viewer with a better understanding of the area being recorded, as well as some understanding of the motion of the camera. In addition to this large static image, a video version was also created, with each frame showing the corresponding frame of the original video, transformed and overlaid onto the large image.

The large static image is created by loading each frame of the original video in sequence, transforming the frame with its respective global transformation, then adding it to a preallocated array large enough to contain the final image. Where one pixel is present in many frames, only the first instance of the pixel being used is desired. This is to show what is as close to a snapshot of the initial

conditions of an area as possible. To this end, a separate preallocated array is also maintained, with the same size as the image array, except only containing logical values. This array has a value of 0 wherever an image has been added to the image array and 1 everywhere else. When adding the next frame, the transformed image is therefore multiplied by the logical array before being added to the image array, after which the logical array is edited to reflect the new occupied pixels. To conserve memory, each frame is discarded after being used.

In order to create the video version of the map, the above process is applied to generate the first frame. The initial image array is the end result of the first algorithm, but the logical is computed differently. Instead of showing the available space in the image array, it show the pixels which are not being used in each frame. The final image array for each frame in the output video is therefore transformed_source + previous_output_frame × logical_array. The end result of this process is that each pixel originally shows its first value, can change value when the camera passes over the area, then is left at its final value after the camera moves on.

2.3.2 Mache stiching

Typically, image stitching algorithms find the minimum number of frames required to cover an area. This reduces the number of frame boundaries and therefore reduces visual artifacts. Minimising the frames in this way loses the camera path information which is useful for analysis. Also, our method has unavoidable cumulative distortions over time; infrequent frame stitching produces very noticeable artifacts at the boundaries. Alternatively, large number of stitchings is clearly not ideal due to many distorted boundary lines.

Mache stitching is a simple method to compromise between the two that we developed. The algorithm has a threshold that controls the frequency of frame stitches. The threshold r is bounded by $0 \leq r \leq 1$, a higher value results in more stitches. For our tests we set $r = 0.8$.

Start with a blank canvas
Paint first frame onto the canvas
Record the position of the corners of the initial frame
for all Frame **do**
  Transform corners using M_t^G into global coordinates
  Calculate intersection between frame and last painted frame
  Calculate ratio of intersecting area to frame area
  if $Ratio > Threshold$ **then**
    Warp frame to global coordinates
    Trim the edges by erosion with a 5×5 square structuring element {This removes black banding at the edge due to interpolation}
    Paint frame onto the canvas
    Record the position of the corners of the frame
  end if
end for

An initial issue with this implementation is memory usage due to the high

resolutions of the final image. It is impractical to handle all stabilized frames in memory. This issue was resolved by implementing the algorithm with lazy evaluation.

2.4 Evaluation

We tested our developed algorithm in three stages. First, we generated a video tracking across a surface of noise to test the ideal case. A background was generated by an edge detected Perlin noise [6] with a color correction. This noise was chosen as it has distinctive corner features for the algorithm to track. A video was produced where the video tracked a 250×250 window along the line parameterized by

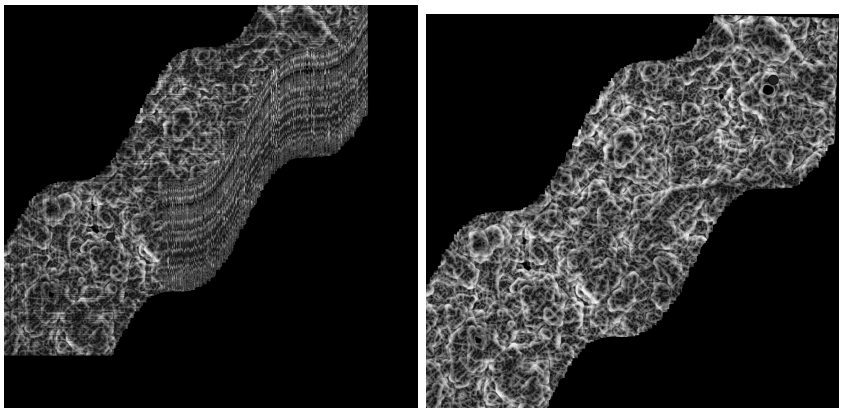
$$y = 50 + 50 \sin\left(\frac{t}{50}\right) + t \tag{4}$$

$$x = t \tag{5}$$

Small random jitters acting by scaling and translating the window was also applied. When saving the video, we upscaled the window to 500×500 to reduce dataloss through video encoding. This path was chosen as it is simple and asymmetric under reflections of the axes; the path is asymmetric to avoid errors 'cancelling' out when moving up and down. Two points were chosen, one in the initial frame and one in the final frame. The distance between the points were calculated in the background image. We applied the algorithm to the video and measured the distance based on the panorama image. The distance of the panorama image was divided by two to correct for the upscaling.

The algorithm was then tested on a tracking across the measuring tape. The accuracy of the algorithm in practice was tested by comparison of estimated spatial distance between known separations. A measuring tape was extended on a table surface. Miscellaneous items were placed around the measuring tape to provide the algorithm with feature points to track. A Samsung Galaxy J5 SM-J500FN was used as the camera. The camera was positioned above the tape such that a small region of the measuring tape was visible. The camera was tracked along the measuring tape with small motions orthogonal to the tracking direction. Our algorithm was applied to the recorded video to produce a panorama. The true distance between the ticks and the zeroth tick is found by simply reading off the value. The pixel to length ratio was found by measurement of the pixel length between two fixed point on the tape in a single frame. The pixel distance of the ticks was measured in the panorama image. The pixel distance could then be converted to a real distance. The absolute deviation of the panorama distance from the true value is plotted.

UAV footage taken over the University of Nottingham School of Physics and Astronomy was provided for our testing. UAV footage was stabilised and visually compared with an aerial image. Bing maps was used as a reference as they provided the latest image of the area. We were unable to find a recent aerial image that included the renovations to George Green Library.



(a) The noise visualised by in-place frame stitching at frame 150

(b) The noise visualised by mache stitching

Figure 1: The camera is tracked along a path through the simulated background. The visual artifact of the in-frame stitching is caused due to interpolation during transformations. Very few artifacts are observed along the path using the mache stitching visualisation.

3 Results

3.1 Simulated data

The stiched output of the simulated data by in-frame stitching is shown in figure 1a, and by mache stitching in figure 1b. The distance measured is between center of the black blobs in the top right and bottom left corners. The distance between the two points in the original image is 800 pixels. The distance between the two points calculated from the video is 1647 pixels. Correcting for upscaling, the calculated value is 824 pixels. This is a deviation of 3% after travelling 3 times the window size.

3.2 Measuring tape

Shown in figure 2. The pixel to meters ratio is 1266pixcm^{-1}. A plot of the calculated distance against the true distance is shown in figure 3. Figure 4 shows the absolute deviation from the expected value. The deviation is less than 1 cm from the true value up to around the 50cm point. A sharp deviation is observed beyond the 50cm point, corresponding to a discontinuity in the image.

3.3 Drone footage

The panoramic drone footage is shown in figure 5a. Minimal artifacts between the frames can be seen in the lower half of the image. The camera is observed

Figure 2: The panorama image produced from a video tracking along the measuring tape using mache stiching. Prior to passing the 50cm mark, the algorithm is reasonably accurate. The discontinuity occurs around the $50cm$ mark due to significantly less surrounding features than the region before it. This is an important result as it demonstrates the conditions needed for the algorithm to become inaccurate.

to shrink as it passes over tall buildings, producing distortions in the upper half of the image.

4 Discussion

4.1 Simulated data

The simulated data was used to test the correctness of the in-frame stitching algorithm, the result is shown in figure 1a. We observe that when a frame moves out of an area, there is a chance that it would leave behind a line of dark pixels, rather than the image which should be displayed. When transformations are applied to the frames, a small number gain a dark, but not completely black, border due to interpolation when the transformation is applied. The algorithm assumes that any areas with a brightness of 0 are outside of all image data, and can therefore be replaced. We resolve this problem with mache stitching where the border is eroded in figure 1b.

4.2 Measuring tape

As seen in figure 3, the software was able to correctly maintain the scale of a recorded object but only up to a certain point. All points below $50cm$ are scaled correctly yet there seems to be some minor discontinuities before $60cm$ is reached. Figure 2 shows these discontinuities clearly. Near the right-hand side of the image there is a point where the image of the tape becomes disconnected with itself, resulting in a difference of angle in the right-most frames compared to the rest of the image. The discontinuity is caused by a vertical gap in which very little features are available except for the measuring tape. This would result in the ORB algorithm finding few feature points outside of the tape. This is a problem as most of these remaining points are situated on the identical ticks on the tape's scale, making it difficult to accurately match each point to itself in neighbouring frames. This shows that when performing video stabilisation, it is

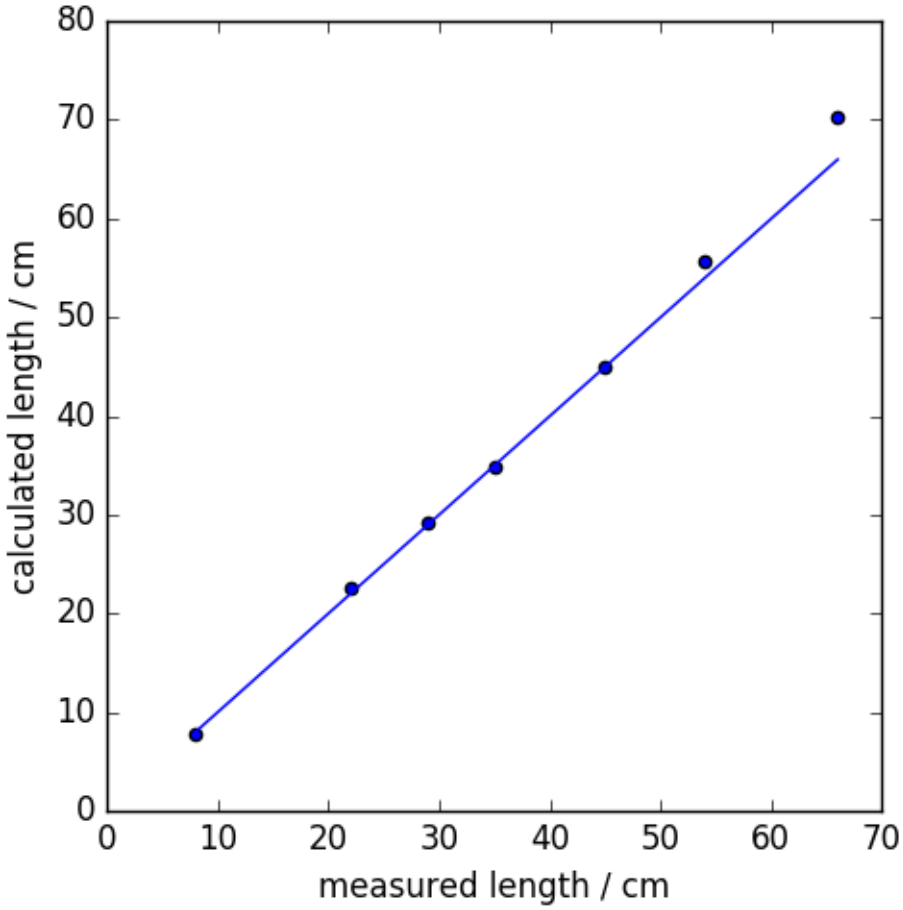


Figure 3: A plot of the calculated length against the known true length. The algorithm is produces a reasonably good tracking as long as many features are available to track.

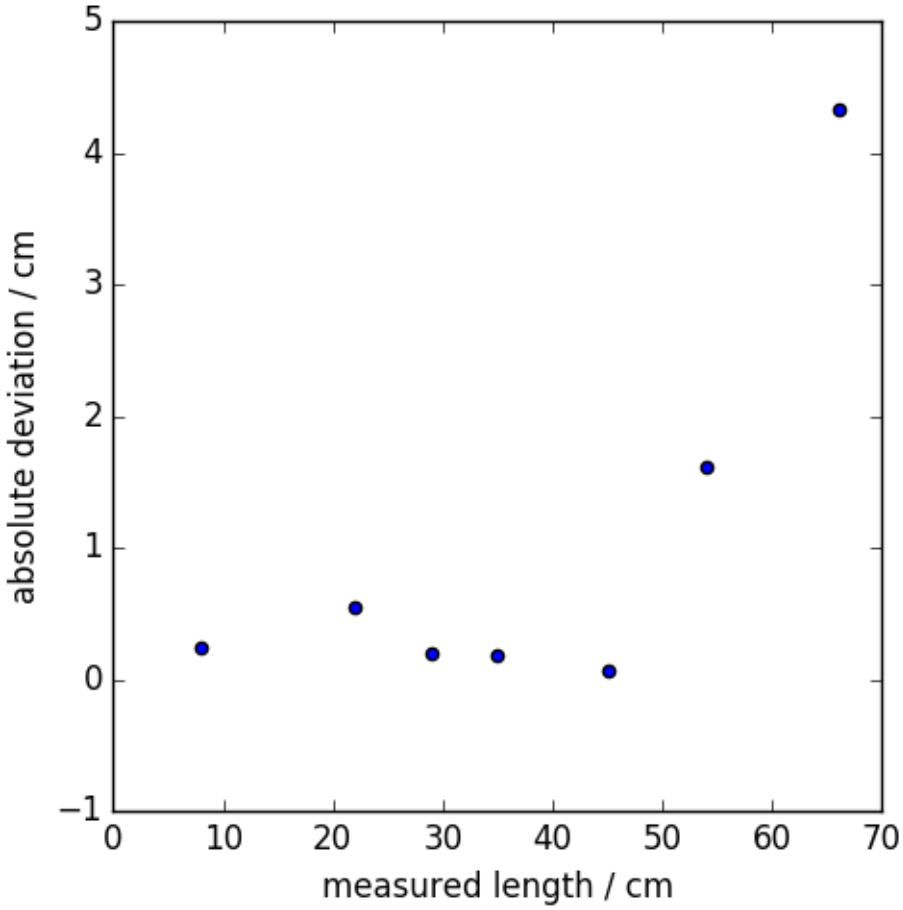
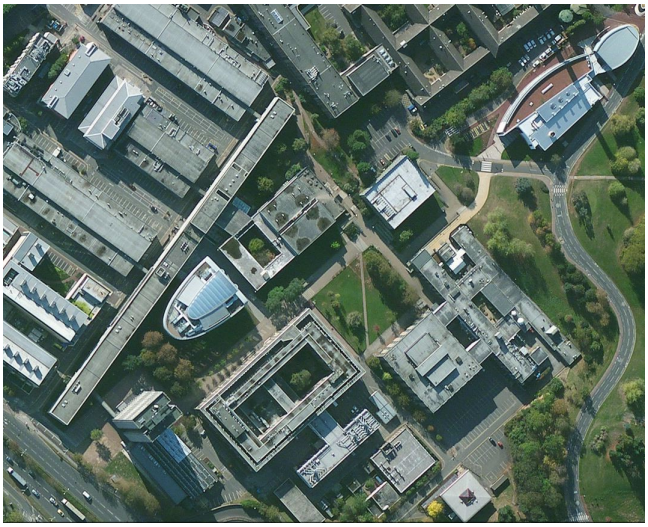


Figure 4: A plot of the deviation of the calculated distance from the expected value. The sharp deviation in this image corresponds to the discontinuity in the image caused by lack of features.

(a) Stabilised drone footage with mache stitching



(b) Aerial view from Bing maps

Figure 5: An aerial view of the same location by two different methods. The algorithm is seen to work very well in the lower half of the image. Heavy distortions occur as the camera moves upwards.

important to ensure that there are sufficient objects which exist in both frames before attempting to calculate transformations, otherwise artefacts are highly likely.

4.3 Drone footage

Finally, the software was tested using real-world drone footage. The video consisted of a tracking shot which starts out over a car park, then moves over a building. Initially, as seen in the lower section of figure 5a, the process works nearly as intended, with only minor artefacts. After passing over a building, however, major inconsistencies start to occur. The most obvious case is the area above the courtyard in the centre of the image, where we see a copy of the courtyard and the buildings surrounding it. Further inconsistencies may be spotted by using real satellite imagery of the area, provided in figure 5b. Of particular note is that although many buildings in the upper section have incorrect locations and scales, their rotations seem mostly unaffected. It appears that each time the drone moves over a building, the displayed frame size is reduced. This is most likely due to the changing size of the buildings' walls due to parallax. As the drone passes over each building, it walls would seem to shrink upon approach, disappear whilst the drone is directly ahead, then expand again once it passes over the other side. The effect of the shrink is small when flying over the car park which show the algorithm will work as long as surface is low-laying.

5 Conclusion

We demonstrated how feature point matching can be used to perform video stabilisation. The algorithm produces smooth videos and it was demonstrated that it also serves as a possible method for generating a panoramic images. The methods presented here were shown to have moderate success in stitching UAV image of low-laying land. Problems were encountered in edge cases where not enough features are present, or where parallax is a significant concern. The analysis conducted is a useful first step in the process of developing software which may help emergency services monitor those in danger, geographical surveys, or better visualise the path taken by objects of interest.

References

[1] S. Kulkarni, D.S Bormane, and S.L Nalbalwar. "Video Stabilization Using Feature Point Matching". In: *Journal of Physics: Conference Series* 787.1 (2017), p. 012017. URL: http://stacks.iop.org/1742-6596/787/i=1/a=012017.

[2] E. Rublee et al. “ORB: An efficient alternative to SIFT or SURF”. In: *2011 International Conference on Computer Vision*. Nov. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[3] E. Karami, S. Prasad, and M. S. Shehata. “Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images”. In: *CoRR* abs/1710.02726 (2017). arXiv: 1710.02726. URL: http://arxiv.org/abs/1710.02726.

[4] D. G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[5] Martin A Fischler and Robert C Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[6] K. Perlin. “An Image Synthesizer”. In: *SIGGRAPH Comput. Graph.* 19.3 (July 1985), pp. 287–296. ISSN: 0097-8930. DOI: 10.1145/325165.325247. URL: http://doi.acm.org/10.1145/325165.325247.