Video Frame Extraction for 3D Reconstruction

Louis G. Clift, Adrian F. Clark
School of Computer Science and Electronic Engineering
University of Essex
Email: {lclift, alien}@essex.ac.uk

***Abstract*—This paper addresses the problem of using live and pre-recorded video sources in 3D reconstruction systems. Typical photogrammetry packages use images generated from digital photography cameras such as DSLRs. 3D reconstruction algorithms require the intrinsic information to be stored in the EXIF header of a digital photograph. Live video systems don't store this information and more importantly don't produce a concise sequence of distinct frames. This paper presents an approach to using both live and pre-recorded video sources with 3D Structure from Motion systems through the implementation of a pre-processor. This approach filters the video feed with the aid of the Laplacian operator and feature-based methods to create an optimal sequence for both on-line and off-line 3D reconstruction systems.**

Keywords—Image processing, video filtering, 3D reconstruction

I. Introduction

Digitising the real-world is a key interest across many fields of study. Capturing the world around us is an important task for both documenting change as well as understanding the environment. Sensing the environment has a wide spectrum of application and is a fundamental component of robotic systems. The problem is at first, well defined, to sense the environment around the platform, yet when it comes to producing a 3D reconstruction this problem significantly increases in complexity. There are a range of approaches commonly employed by robotic inspection systems which provide an extremely high level of precision and accuracy such as LiDAR, CAT or MRI solutions, however these solutions are extremely expensive and immobile (with exception to LiDAR) restricting their availability and practical application for mobile robotics.

The rise of 3D interactive-gaming has led to a range of RGB-D depth sensors being available to consumers since the release of the original Microsoft Xbox Kinect (Kinect for Xbox306). An RGB-D sensor comprises of an Infra-red projector and IR-sensor alongside a standard (RGB) camera. An RGB-D sensor can be used to sweep over an object recovering both the structure and appearance of the target in near real-time [1], [2]. RGB-D reconstructions lose the sensor location without the aid of external sensors such as an IMU or accelerometers. Although this technique offers a fast method of reconstructing an object, these devices are restricted by the need to be tethered to a computer thus limiting their use in hard-to-reach areas or on robotic platforms. Robotic platforms will typically feature at least one on-board camera which can be used for remote inspection, tracking, object detection and other vision-based tasks.

Vision-based 3D reconstructions offer the ability for users to utilise relatively low-cost consumer-grade hardware to achieve high-quality 3D models through the use of standard camera systems, including those found on a robot. The most common approach is to use two or more cameras, known as *computational stereo* for two cameras or *multi-view reconstruction* for many [3]. Similar approaches can also be applied using a single (monocular) camera using a technique known as Structure from Motion (SfM). Armed with just the image sequences generated from the single camera and their respective meta-data, the 3D SfM algorithms are able to recover both the structure and appearance of the scene as well as an estimate of the original camera positions.

This work is motivated by the wish to develop a technology that allows 3D shape to be captured in either indoor or outdoor environments using a video sequence generated from a remote device, rather than traditional still frame selected by an operator. The intention is to receive a live video stream whether that be from a hand-held camera, mobile device, remote controlled aircraft or a robotic platform; and be able to generate a core subset of frames that would be suitable for reconstruction using a form of Structure from Motion [4]. Structure from Motion is an extremely computationally expensive task with reconstructions taking up to several weeks depending on the number of frames, format and the processing hardware. Therefore selecting the right subset of images is crucial to eliminating any unnecessary processing and keeping processing times down to a minimum.

The solution proposed in this paper is aimed at filtering and preselecting images from video sources specifically for Structure from Motion reconstruction pipelines. The pre-processor is used to filter a video sequence or live stream in order to generate a collection of frames which are free from blur and overlap by a specified threshold. Providing Structure from Motion with too many identical frames (or frames which have a high percentage overlap of greater than 90%) has a significant detrimental impact on required time to produce a model. A video sequence will contain several frames where there is little to no change between frames due to the higher capture rates and slow (or no) motion of the camera. The filtering system ensures that the frames are only passed onto the reconstruction pipeline if they provide a suitable baseline. The system also produces and stores the SIFT features for each frame as a by-product of the overlap analysis stage and removes the feature detection / extraction phase from the Structure from Motion tool-chains.

The remainder of this paper is structured as follows; Section II describes the principles of Structure from Motion, the system architecture and where this work fits into the global processing pipeline. This is followed by a detailed explanation of the image analysis stage in section III. Section IV describes the feature-based selection approach and discusses the use of GPU processing to accelerate the sub-processes. Followed by a series of tests used to evaluate the approach presented in section V. Finally, section VI draws conclusions and identifies areas for further work.

II. System Overview

In-order to use images for 3D reconstruction there are a set of core requirements which the capture method must take into account. Raw video is typically unusable for Structure from Motion due to the large volume of frames in-between the 'correct' shots. The typical capture method for Structure from Motion is to shoot individual frames with a digital camera which encodes the focal length directly into the image. The photos need to be captured at a steady pace whilst moving around the target object (or environment) whilst keeping the camera facing inward towards the centre of rotation. Structure from Motion is a computationally expensive task due to the vast number of calculations which are required to compute both the motion and the depth. Every image is pair-wise matched against the other images in the dataset to identify overlapping viewpoints from each of the 'cameras'. Structure from Motion is able to use the same principles of multi-view stereo, whether there are multiple cameras or a single camera moving through a scene. There needs to be a discrete baseline between two images looking at the same point of interest for Structure from Motion (and other 3D techniques) to be able to recover the depth of the 3D point. If the baseline is too small there will be more processing work due to greater number of images covering the same area. Additionally if the images are blurred then the feature matching process is likely to produce erroneous matches and effect the whole model when the point-cloud is adjusted.

From this criteria it can be seen that there are two key areas which need to be addressed when converting a video sequence into frames for reconstruction; distortion and frame overlap. The image analysis is performed on each incoming frame as this is the lightest component with respect to the computational cost involved. The image analysis stage rejects any frames which have become distorted through soft focus or rapid motion leading to motion-blur. Section III provides a detailed explanation of the algorithms involved with detecting blurry frames. The next phase of the pipeline is to extract the key features within the frame and compare it to the previous (successful) frame in order to measure the overlap. This stage will reject any frames where the percentage overlap is too great. It may seem counter-intuitive at first, however if the frames are too similar then the camera has not moved far enough for the next frame to be sent to the reconstruction algorithm. The images need to have between 60-80% overlap to be suitable for saving. This stage serves two purposes, the first being to reject unnecessary frames but also to reduce the processing stages within the Structure from Motion pipeline.

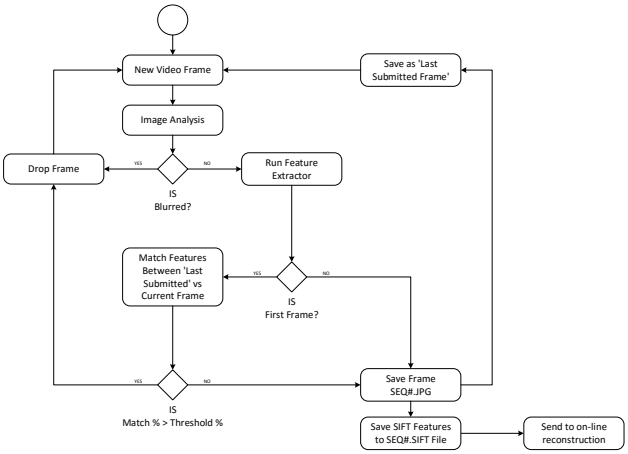

Fig. 1: Video Sub-Sampling System Overview

III. Image Analysis

Detecting the focus of an image is a well-established problem within the field of computer vision which is most commonly found in use in auto-focus systems. Digital cameras utilise a range of methods [5] to analyse a frame and adjust the focal length of the lens until the target area comes into focus. Professional digital cameras offer fast focus detection to reduce the time required to capture the photograph before the opportunity is missed. Structure from Motion is sensitive to distorted images, as blurred images lead to a loss in reliability and uncertain matching. When capturing frames by hand a photographer would typically wait until the scene is in focus before (and after) capturing the photo which produces a sequence of clear and accurate images for use in the reconstruction system. Video cameras are more challenging to work with as the frames are not all perfectly focused. Working with the assumption that the video source is progressively scanned, one can assume that each frame of the video sequence is a complete image. However, if every frame is extracted there will still be cases where there is motion-blur even though the camera is in focus when stationary. The capture rate of the sensor (and recording hardware) is key to the level of blur seen in the final frames. A higher recording rate will enable the camera to avoid blurring in fast movements. This can be observed by moving a camera in a fast arc whilst taking a photo. Lower-quality devices such as web-cameras and sub-720p streams normally experience high levels of motion blurring.

To protect the 3D reconstruction system from blurred frames the incoming video sequence (live or pre-recorded) is analysed frame-by-frame and evaluated against a *blurriness* threshold. The score is determined by first converting the frame to grey-scale and then calculating the variance of the Laplacian [6]. The Laplacian operator (often referred to as LoG, Laplacian of Gaussian) is used to find the edges in the image. Edges which are sharp and in focus will present higher contrast than

blurred edges. The Laplacian operation outputs a frame with just the edges present. The variance of the resulting frame is then calculated returning a single score for the level of blur in the image. A sharp, focused image will have a high level of variance where as a soft or motion-blurred image will have a low amount of variance. The variance or 'blurriness' threshold varies depending on the capture source.

There are multiple ways to generate the variance of a given array, with the benchmark method being the 'Two-pass' method:

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})}{n-1} \tag{1}$$

The problem with this method is that the algorithm requires two passes over the array which is computationally expensive. A two-pass variance calculation takes 29ms to execute on a 1920px x 1080px (Full HD) image using an Intel i7 processor. The total time required for the Laplacian operation and calculating the variance clocks in at 42.5ms per frame. Detecting blurred frames using this method would be unsuitable for live video captured at 25 fps as the stream produces a new frame every 40ms. Before experimenting with the method of calculating variance it's key to note that the Laplacian operator requires 13.5ms to complete on the same HD frame, therefore the ideal time to compute the variance would be below 10ms.

The aim is to calculate the variance in a single pass through the array. Switching from the two-pass to the 'on-line' method significantly reduces the computation time to just 18ms. The on-line method (shown in equation 2) produces a running estimate of the variance until the end of the array is reached where the resulting variance matches that of the two-pass method.

$$\begin{aligned} M_{2,n} &= M_{2,n-1} + (x_n - \bar{x}_{n-1})(x_n - \bar{x}_n) \\ s_n^2 &= \tfrac{M_{2,n}}{n-1} \end{aligned} \tag{2}$$

Although the time to calculate the variance has been lowered to a suitable speed, the timings can be further improved by adjusting the sampling method. The current approaches calculate the variance for the entire image. Given that the blur will exist across the whole image (with the exception of depth-of-field effects) one can sample the image to reduce the number of pixels that require processing. Figure 2 shows the effects of sampling the image at a range of intervals whilst measuring the processing time and variance of the results compared with the output from the two-pass method. The results shown in Figure 2 excludes the performance of the two-pass and on-line methods. An image is a 2D matrix, therefore the sampling methods each pick points based on a row-skip and column-skip parameter. The complete range from 1,1 (online) through to 7,7 are evaluated to explore the speed and error of each of these sampling window sizes. There is a clear trade-off between execution time and the error. Although this is a fairly intuitive observation, what is interesting is the sampling method used to calculate the variance. In the interest of accuracy an upper bound of 4% difference has been applied when analysing the methods. Table I shows the sampling methods within the region of $2\% - 4\%$ sorted by their percentage difference from the variance result produced by the two-pass method. Sampling pixels at every 2^{nd} row and every 3^{rd} column or vice-versa produces results in the fastest time within this bracket. However the results from these two methods produce very different results. S15 preforms worse than S9 implying that skipping more rows than columns has a detrimental effect on calculating the variance of a Lapalcian image.

TABLE I: Sampling Methods: 2-4% Difference

Method (row, col)	Difference (%)	Time (ms)
S8: 2,2	2.37215	5.40776
S14: 3,1	2.62616	7.20041
S9: 2,3	2.72382	3.6048
S3: 1,4	3.44564	5.40285
S15: 3,2	3.55667	3.60128
S4: 1,5	3.5687	4.32103

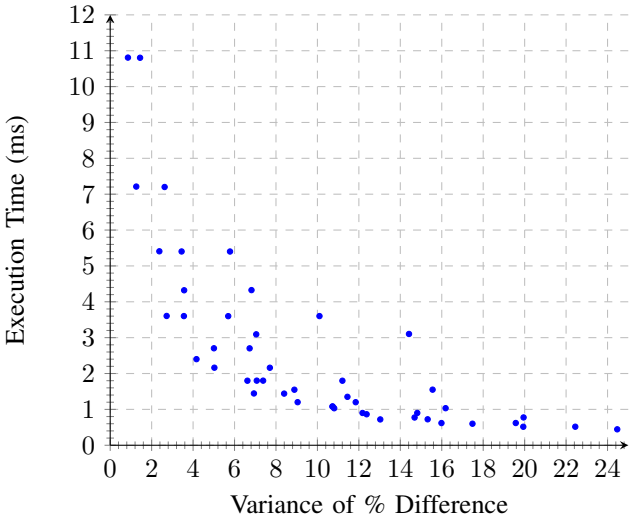

Fig. 2: Variance By Sampling: Execution Time vs Error

Based on the results from these experiments, sampling method S9 (2,3) has been selected for use in the rest of this work. S9 reduces the blur detection process time down to just 16.3ms (including the Laplacian operation) on a Full HD frame without a significant impact on accuracy. The video preprocessing pipeline should be capable of working with a range of video sources including low-resolution images which are more common on live robotic platforms where bandwidth is limited.

IV. Frame Selection

Once the image analysis component has determined that the frame is relatively distortion-free it can be passed into the next stage of the pre-processor, frame selection. A 25 fps (PAL) progressive video sequence will generate a new frame every 40ms which generates a vast number of redundant frames, significantly affecting reconstruction systems. A series of identical or very similar frames are unsuitable for use

in reconstruction due to the marginal displacement between the images. This is especially visible when a vehicle stops or remains stationary for any length of time. To protect the reconstruction pipeline from excessive processing this system analyses the incoming frames in a pair-wise manner, similar to the reconstruction algorithms, and rejects any frames which have a high portion of overlap.

A. Feature-Based Analysis

Typical reconstruction algorithms [7], [8] will use SIFT or SURF as the primary feature detector / extractor for analysing the input images. Both of these feature detectors are computationally expensive taking several seconds to complete on a high-end mobile workstation. GPGPU, General Purpose Graphics Processing Units, have been able to significantly reduce this time down into the milliseconds. In keeping with the goal of developing a fast (near) real-time pre-processor a GPU SIFT library [9] has been experimented with, alongside other feature detectors to extract the key points from the incoming frames. The GPU can be used for both extraction and pair-wise matching, thus speeding up a complex task which would otherwise rely on the CPU for. Once a set of features and the corresponding matches have been calculated the system then compares the frames for overlap.

The distribution of features between frames in the overlapping region of pairs of images is key to the accuracy of the reconstruction process, as discussed in a recent study [10]. The authors concluded that SFOP is the best performing feature detector for overall key point coverage in overlapping areas, which is an important attribute for stitching and 3D reconstruction systems. SIFT comes third (behind IBR). Both SIFT and SFOP have GPU implementations, however at the time of writing this paper the author was unable to implement the GPU-variant of SFOP to gather competing results. SFOP generates 10,000 key points from each of the input images taking up to 25 seconds per frame (22 average). SIFTGPU generated less keypoints (1600) for the same sequence, however completes in 100–175ms per frame, a significant improvement.

These experiments revealed that real-time SIFT for high resolution imagery may still be out of reach using the given libraries so far. Measuring the overlap between frames does not require high levels of accuracy, therefore we can afford to switch to a faster feature detector in the interests of a deploying an on-line frame extractor. OpenCV features an extremely fast feature detector, ORB (Oriented FAST and Rotated BRIEF) [11], which has both CPU and GPU implementations. ORB CPU produces approximately 2,500 key-points per frame in just 16.8ms, significantly faster than SIFTGPU. Interestingly the GPU implementation is slower for ORB[1] than the CPU version due to upload/download to GPU. ORB GPU produces the same number of features as the CPU version however it requires 24 – 29ms. Given the speed of the CPU implementation the final pre-processing system is able to achieve video-rate frame filtering. (Blur analysis: 16.3 ms + ORB detection: 16.8 ms = 33.1 ms). These findings are initially confusing, however once the feature extractor is added into the system the GPU implementation remains at 29.46ms whereas the CPU time increases to 40.65ms.

[1]Experiments run with OpenCV versions 2.4.10 & 2.4.13

B. Overlap Analysis

Analysing the overlap between two frames is the core function of this work. The frames need to be sufficiently different yet provide enough overlap for Structure from Motion. The first non-distorted frame is used as the initial reference image. A Brute-force (BF) Feature matcher is used to find the corresponding matches between the reference image and the current frame from the video source. The CPU-based Brute-force matcher is the bottle-neck of the system. The matcher takes upto 550ms to compute based on 2,500 matched pairs. A GPU Brute Force matcher is crucial to recovering the time. The GPU implementation takes 25 – 80ms to compute the same frames as the CPU variant.

A metric for similarity is obtained by calculating the distance between the matching key-points. The feature-detector determines which distance measure is used. Floating-point feature descriptors such as SIFT and SURF use the Euclidean distance measure, whereas binary feature descriptors such as ORB require the Hamming distance measure. A threshold is used to determine whether a frame should be kept or discarded based on the distance between the keypoint pairs. If the features are greater than the given threshold then the frame is used as the reference frame for cross-comparison against the new frames. Section V details the threshold selection. The aim is to set the threshold such that there is 60% overlap between frames from the incoming sequence.

C. Exporting Keypoints

Once two frames satisfy the overlap threshold, the frames are stored. The stored frames are processed by a worker thread which uses SIFT to generate a feature descriptor file. The SIFT files are used by reconstruction pipelines (Bundler, Visual SFM), therefore the SIFT results can be reused which reduces the workload for reconstruction systems. The feature writer has a longer cycle period, especially during slow manoeuvres where the difference between frames is less. Therefore the SIFTGPU module has enough time to execute.

V. Results

A test sequence has been created using a ground robot driving around a cluttered indoor environment. A GoPro camera was fixed to the vehicle, mounted 45° towards the centre of the room. The robot drives around the room capturing a video sequence at 50fps. The video sequence was filmed in full high-definition, generating a total of 7,850 frames (157s). The frames often include blur due to the vibration and the fast rotation of the robot. An extraction rate of 2 frame per second would create 314 image, although this is a reasonable number of frames the images do not provide ideal visual coverage for reconstruction.

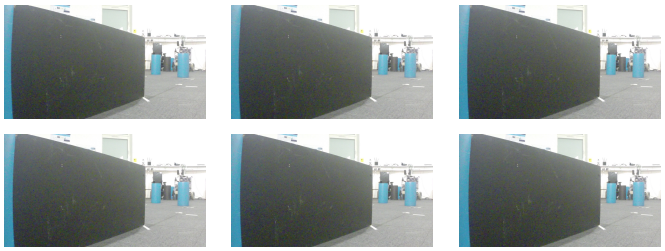

Fig. 3: Threshold = 10, Total Output Frames = 4879

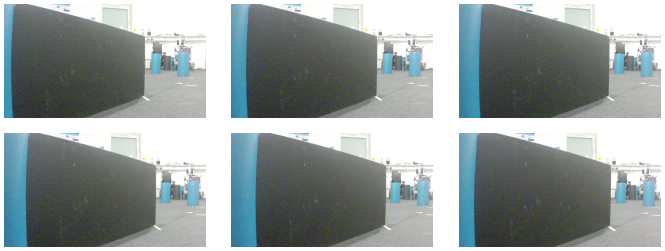

Fig. 4: Threshold = 25, Total Output Frames = 841

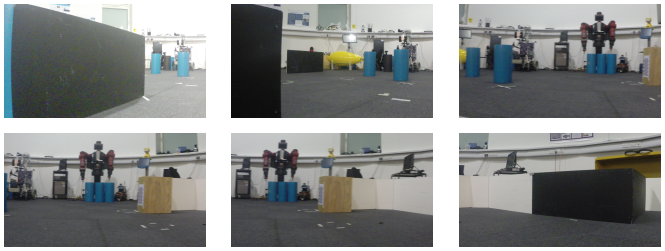

Fig. 5: Threshold = 30, Total Output Frames = 241

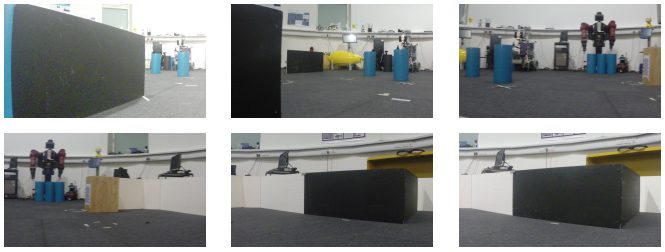

Fig. 6: Threshold = 35, Total Output Frames = 116

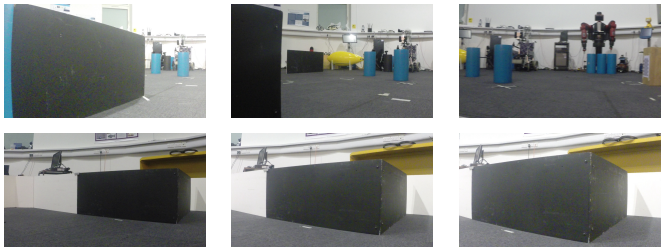

Fig. 7: Threshold = 40, Total Output Frames = 68

Figures 3–7 show the first 6 frames selected for reconstruction using the approach developed in this paper. Each of the figures shows the output based on the key-point distance threshold. The results show a sharp descent in the number of images produced. A Hamming distance of 40 produces a significant jump, missing a lot of intermediate detail and reducing the dataset to just 68 images as seen in Figure 7. On the other end of the scale, a distance of 10 causes excessive overlap. The frames seen in Figure 3 show almost identical images. The video sequence is an extremely challenging task for the 3D reconstruction pipeline due to the camera being positioned at an angle. The camera is set to provide a trade-off between forward observation and capturing the target environment.

Fig. 8: Sparse 3D Model of the Arena using t=25

To analyse the performance with a more realistic input feed, a hand-held video was captured of an outdoor seating area. This time the video was captured from a smart phone. The second test video sequence is a recording of a circuit of the University of Essex Colchester campus squares. The camera is aimed towards the centre of the square to follow a normal Structure from Motion capture path. The video was captured at a medium paced walk lasting 1 minute 17 seconds, producing 2,310 HD frames. Starting with the distance threshold (t) set to 25, the preprocessor created 1,773 frames, whereas a threshold of 55 yields a more manageable 967.

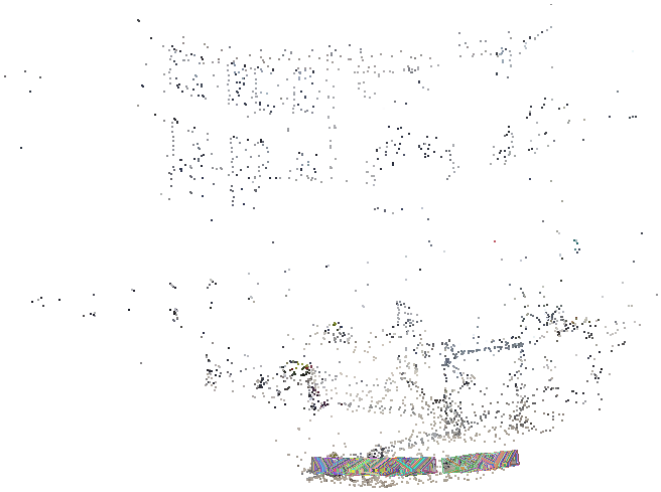

Fig. 9: Sparse 3D Model of Square 2 using t=55

The sequences used so far are generated from a relatively stable ground vehicle and a typical path. The direction of motion will be in one of four directions, rotation to the left, rotation to the right or forwards and reverse. A more challenging task would be to use video from a UAV such as a Parrot Bebop. A UAV has 9 degrees of freedom allowing for more complex motion of the camera. The Bebop features a Full 1080p stabilised camera. A flight was recorded of the aircraft spinning up for take-off until the aircraft reaches the top of a castle (25m) taking just 20 seconds. The images contain a significantly higher number of distinct features, reaching the 10,000 limit for ORB. Applying a threshold of 25 form the arena dataset produced the same results as those seen in Figure 3. A series of trials revealed that a Hamming distance of 50-55 produces a suitable set of images for use in reconstruction as shown in Figure 10. When the threshold (t) is set at 25, the system creates 564 images whereas t=50 generates 524, and t=55 generates 431 images from the 20 second clip.

Fig. 10: Aircraft Take-Off with a Threshold of 50

VI. Conclusions

The principal aim of this work was to develop a utility that enables live video feeds to be used in both existing and new 3D reconstruction systems. Video produces an additional set of challenges for Structure from Motion which this work looks to solve & reduce. The pre-processing is unable to run at video-rate using the feature detectors in use by 3D reconstruction systems, however this work and others have shown that alternative feature detectors can address this timing issue and bring 3D Structure from Motion ever closer to live reconstruction. The solution presented in this paper requires two threshold values (blurriness & distance) to be adjusted based on the video source. A distance threshold of 25 worked well for the GoPro camera, however a much higher threshold was required to achieve the same results with the hand-held and video from a UAV.

The total frame-wise time cost using this approach is 70.76ms – 125.76ms (depending on number of key-points detected). The development of this system has highlighted that the major bottle-neck lies in the matching strategy. Further experimentation and investigations into feature matching would be required in order to reduce the overall processing time of this pre-filter. The current system is capable of running at 14Hz, short of the 25Hz required for live video-rate processing.

Fig. 11: Sparse 3D Reconstruction from 20 seconds (t=55)

Acknowledgements

The first author acknowledges the receipt of EPSRC doctoral training funding to support his work.

References

[1] D. Alexiadis, D. Zarpalas, and P. Daras, "Real-time, full 3-d reconstruction of moving foreground objects from multiple consumer depth cameras," *Multimedia, IEEE Transactions on*, vol. 15, no. 2, pp. 339–358, 2013.

[2] D. Alexiadis, G. Kordelas, K. C. Apostolakis, J. D. Agapito, J. Vegas, E. Izquierdo, and P. Daras, "Reconstruction for 3d immersive virtual environments," in *Image Analysis for Multimedia Interactive Services (WIAMIS), 2012 13th International Workshop on*, 2012, pp. 1–4.

[3] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, Mar. 2004.

[4] K. Häming and G. Peters, ""the structure-from-motion reconstruction pipeline — a survey with focus on short image sequences," *Kybernetika*, 2010.

[5] S. Pertuz, D. Puig, and M. A. Garcia, "Analysis of focus measure operators for shape-from-focus," *Pattern Recognition*, vol. 46, no. 5, pp. 1415 – 1432, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320312004736

[6] J. L. Pech-Pacheco, G. Cristóbal, J. Chamorro-Martinez, and J. Fernández-Valdivia, "Diatom autofocusing in brightfield microscopy: a comparative study," in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 3. IEEE, 2000, pp. 314–317.

[7] Changchang Wu, "VisualSFM: A Visual Structure from Motion System." [Online]. Available: http://homes.cs.washington.edu/~ccwu/vsfm/

[8] N. Snavely, "Bundler: Structure from Motion (SfM) for Unordered Image Collections," Online. [Online]. Available: http://www.cs.cornell.edu/~snavely/bundler/

[9] Changchang Wu, ""SiftGPU: A GPU implementation of Scale Invaraint Feature Transform (SIFT)"." [Online]. Available: http://cs.unc.edu/~ccwu/siftgpu

[10] E. Bostanci, N. Kanwal, and A. F. Clark, "Spatial statistics of image features for performance comparison," *IEEE Transactions on Image Processing*, vol. 23, no. 1, pp. 153–162, Jan 2014.

[11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: an efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2564–2571.