Australian National University

Evaluation of Visual Object Tracking Algorithms on SPAD Camera Data

October 27, 2017

Jack Henderson

u5561978

ENGN3712 - 12 Unit R&D Project

Supervised by

Associate Professor Jochen Trumpf

of the

College of Engineering & Computer Science

Australian National University

in collaboration with

Defence Science and Technology Group

Abstract

We explore the process of visual object tracking on data obtained from Single Photon Avalanche Diode (SPAD) arrays. The task differs from traditional visual object tracking due to the unique characteristics of SPAD data, namely high noise and coarse resolution. The performance of three state-of-the-art visual trackers, STRUCK, KCF, and Meanshift, is evaluated on a series of synthetic sequences, which are created using statistical models of supplied SPAD data. Even with the raw SPAD data as the input, the three trackers perform significantly better than expected, surpassing the performance observed on benchmark sequences captured with traditional digital cameras. Performance can be further increased with the introduction of a newly created image feature, which utilises the temporal characteristics of the SPAD signal. Overall, STRUCK provides the best performance in terms of accuracy and robustness. Our results demonstrate that visual object tracking methods are a viable and promising approach to object tracking on SPAD data.

Acknowledgements

The continued support of my supervisor, Jochen Trumpf, throughout this project has been invaluable. With his guidance and wisdom, I was able to overcome the countless challenges and hurdles this project has thrown at me. He has also challenged me to achieve better and pushed me to the limits of what I thought I was capable of. The result is a project that has delivered on all of its initial aims and a report that I am truly proud of. Neither of these would have been possible without Jochen's unwavering support, and for that I am truly grateful.

I would also like to acknowledge the contributions of the Defence Science and Technology group, and in particular Joyce Mau. The experimental work and data collection performed by Joyce was key to the success of this project as it provides the basis for much of the analysis. I am deeply thankful for Joyce's efforts and the support of DSTG.

Contents

1 Introduction **3**

2 Background **4**

2.1 Single Photon Avalanche Diode Arrays 4

2.2 Object Tracking 4

3 Literature Review **8**

3.1 Point Tracking and Silhouette Methods 8

3.2 Kernel Methods 8

3.3 Machine Learning 9

4 Selection and Implementation of Trackers **11**

5 Analysis on Benchmark Datasets **12**

5.1 Metrics 13

5.2 Parameters and Features 14

5.3 Results 14

5.4 Validation 20

6 SPAD Data Modelling **20**

6.1 SPAD Data Observations 21

6.2 Creating Synthetic SPAD Sequences 23

7 Analysis on SPAD Datasets **25**

7.1 Tracking with raw data 29

7.2 Tracking with peak-count feature 29

7.3 Tracking on DSTG Dataset 31

8 Conclusion **34**

9 Future Work **35**

Appendix A Tracking Algorithm Parameters **38**

Appendix B MATLAB Test Bench Overview **40**

1 Introduction

Visual object tracking is a large field in the computer vision community as it has a wide range of potential applications. Put simply, visual object tracking consists of identifying the position of a target in a sequence of images, thereby providing a track of the target in the image plane over time. A number of different approaches to object tracking, with the most successful and versatile utilising techniques such as kernel methods and machine learning. However, these algorithms are typically evaluated on and tailored towards high resolution sequences that one can obtain from a standard digital camera. A Single Photon Avalanche Diode (SPAD) Camera is a unique type of sensor that captures images by counting individual photons incident to the sensor. It has several promising applications as it is capable of capturing images of extremely distant objects as well as determining distance to targets via laser ranging. The key difference between SPAD camera data and data from standard digital cameras is that SPAD images have very coarse resolution and high levels of noise.

The differences between SPAD sensors and standard cameras make the task of tracking much more challenging and complex, and this is an almost unexplored area of research. With this in mind, the aim of this project is to evaluate whether standard visual object tracking algorithms are a viable approach to object tracking on sequences captured with SPAD sensors. Our results show that this is indeed the case, and performance on SPAD data can be as good as, if not better than, tracking performance on high-resolution sequences.

To come to this conclusion, we first complete a survey of the existing state-of-the-art tracking algorithms to select a representative sample of the current approaches that show the most success. We implement and evaluate these algorithms on a set of benchmark sequences to demonstrate the typical levels of performance that are achievable at present. As a ground truth trajectory of the provided data was not available, we create a statistical model of a SPAD image sequence. This was used to create synthetic datasets with known target trajectories. We also develop a new image feature to better extract the underlying temporal properties of the SPAD sensor data. Comparing the tracking results on the synthetic SPAD dataset to the benchmark sequences, we show that the trackers selected perform very well, with high levels of accuracy and robustness. Finally, we demonstrate the trackers on the original SPAD dataset and observe similar levels of success.

2 Background

2.1 Single Photon Avalanche Diode Arrays

A Single Photon Avalanche Diode is created by reverse biasing a p-n junction above the reverse breakdown voltage. This creates a strong electric field in the depletion region of the junction but at this stage, only the leakage current flows though the diode. If a photon is absorbed into this region, it will ionise an atom and create an electron-hole pair. Due to the strong electric field, the electron and hole travel to opposite sides of the junction at high speed. For sufficiently high voltages, the electric field accelerates the charge carriers to a speed such that impacts with other atoms in the depletion zone cause those atoms to also ionise - a process called impact ionisation. These carriers are also accelerated by the electric field and thus impact and ionise further particles. This creates an avalanche effect whereby one single photon induces a cascade of charge carriers, resulting in a measurable current across the junction [1]–[3].

One advantage that SPADs have over other light sensors is that there is a very small standard deviation in the time between a photon being absorbed and an avalanche current being detected - on the order of 10-100 picoseconds [3]. This makes it ideal for use in Time of Flight (TOF) distance measuring. In TOF measurement, a laser pulse is emitted and the light reflected off the target is detected by the SPAD sensor. The time between the emission and detection can be used to calculate the distance to the target. Using this approach, Niclass et. al [3] are able to achieve an uncertainty of 1.8mm over a distance of 3m.

More recently, research and development has been focussed towards creating arrays of SPAD sensors [3], [4] which can be used to create a depth map of a scene. However, the size of these sensors is currently limited by available technology, with a typical sensor in the range of 1×8 to 32×32 pixels. At the cutting edge, Burri et. al. [5] were able to manufacture a 512 × 128 pixel sensor. Due to the infancy of this technology, the low resolution and high noise characteristics, there has been little focus on object tracking using these types of sensors.

2.2 Object Tracking

Object tracking is one of the primary tasks in computer vision. The goal of an object tracking algorithm is to identify the position of an object, the 'target', over time in a video

sequence of images. Importantly, an object tracking algorithm only provides the position of the target in the image plane, not in 3-D space.

The process of object tracking can be divided into a number of logical steps, namely; object representation, feature selection, object detection, and tracking. The process can also be made more challenging by the consideration of multiple objects, occlusions of the target or varying lighting conditions.

2.2.1 Object Representation

A target can be represented in a number of different ways, and the chosen representation is highly dependent on the type of algorithm being used as well as the target being tracked. Simple representations include a single centroid point, a set of points or a geometric shape [6]. Simple representations are mostly suited to rigid objects as their movement can be described as a simple transformation of the representative target. For example, in a video of a ground vehicle taken from an aerial camera, a rectangular bounding box would be an ideal representation of the target vehicle. The target representation also limits what information the object tracker can provide. For example if a point representation is used only the target's translation is describable, whereas a rectangular model can be used to describe a translation, scale and rotation. We recall that in object tracking, it is sufficient to simply describe the location of the target within the frame - a full pose description is irrelevant.

For more complex targets, additional representations can be used. An object contour or silhouette can be used for targets that are non-rigid or are not represented well by simple geometric shapes. In cases where the target is highly constrained to very specific objects, further information about the object can be leveraged. This is especially prevalent in human body tracking algorithms. In these cases articulated shape models or skeletal models can capture the underlying structure of the target as well as kinematic constraints [7].

2.2.2 Feature Selection

Feature selection is the process of extracting useful information out of the image which is then used in the object detection process. A feature describes to what degree a region of the image expresses a particular property of the scene. For example, transforming the image by taking a derivative will highlight areas of sudden brightness change, possibly

indicating the edge of an object within the scene. Simple features can include colour and brightness, while further processing of the image allows for the identification of more advanced features such as edges, corners, textures, and patterns.

Often, a combination of features will be used for object detection. The most useful set of features are ones that are uncorrelated with each other and are also able to provide a unique representation of the target. An example of this can be seen in Hare et. al [8] where using a combination of Hare-like features and intensity histogram features results in better performance than either alone. As with object representation, the choice of features is highly dependent on the type of objects being tracked and the data that is contained in the image sequence.

There are several desirable properties that make a good feature selection algorithm. It is common in many object tracking scenarios for the object to move closer or further from the camera, causing its apparent size in the image to change. In order for the object to be tracked accurately the features expressed should be similar, irrespective of the apparent size of the target in the image [9]. This is commonly known as scale-invariance. Another critical component is the consistency and robustness of the feature selection algorithm. Features should be identified consistently across multiple frames, and small changes in the appearance of an object should not significantly change the features expressed. One example of where this is a challenge is when the illumination of the target changes, commonly due to shadowing or reflections. Simple features that are not illumination invariant do not account for these changes and can produce inaccurate information for the object detection algorithm. Ultimately, the overall success of a tracking algorithm is highly dependent on the quality of the features selected and consistency and accuracy in which these features are extracted from the image.

2.2.3 Object Detection

Once a set of features has been chosen and computed for a frame in the video sequence, the goal of the object detector is to identify points or regions of interest that may correspond to the target that is being tracked. This can be done through a number of means, including key-point detectors, image segmentation or machine learning. If the description of the target is known in advance, for example in vehicle or pedestrian tracking, then the object detector can be trained on this information to more accurately detect potential targets.

A commonly used method for point detection is Lowe's Scale Invariant Feature Transform (SIFT) [9], which selects a set of key-points from the image and uses a 128 element

feature vector to describe each point. This description is invariant to scale and can be used to compare two SIFT points between frames. As would be expected, this approach is generally suited to tracking algorithms that use point-based representations of the target.

Background subtraction is often used as a technique to detect movement between subsequent frames and thus potential targets. It can also be used to exclude some parts of the image from further processing, reducing the computational power required and decreasing the probability of false matches [10]. However in order to perform background subtraction, either the camera must be stationary or the trajectory of the camera must be known. Also considering the nature of the data that is likely to be obtained from the SPAD camera, especially the low resolution, background subtraction will not be an area of focus.

2.2.4 Tracking

The goal of object tracking is to take the information generated from the object detection step and combine it with the model of the target to locate the target within the current frame and subsequently generate a track of the object over time. Two approaches have historically been taken to object tracking. One approach is to consider all candidate targets in the frame and compare each one with the target model to determine the best match. The other is to assume that the motion of the target is comparatively small, and so the location of the target in the subsequent frame is within a small distance of its current location. This has the advantage of reducing the size of the search space but has the potential to lose the target if the distance travelled between frames is too high. As camera technology improves and frame rates increase, this is becoming less of an issue in most circumstances.

Broadly, there are three different categories that most trackers can be classified into; point tracking, kernel tracking and silhouette tracking. Point tracking uses a key-point detector as described above and matches these key-points between the model and each frame in the sequence. The differences between algorithms in this class relate to how the matching is performed, what assumptions are made about the motion of the target, and which key-point detector is used. In kernel tracking, the target is represented by a 'kernel', normally a geometric region of the image. To locate the object in a subsequent frame, the region of the frame that matches closest with the kernel is identified as the target. This then describes a transformation of the kernel, a combination of a translation, scale, and potentially rotation and shear. Silhouette tracking is used when the shape of the

target can change significantly over time. This can either be done by evolving the target's contour model or by matching the target model with shapes identified in the image.

3 Literature Review

3.1 Point Tracking and Silhouette Methods

Point tracking methods are a popular way to track moving targets in a stationary scene, such as pedestrian tracking from a surveillance camera. A variety of feature detection algorithms have been developed, including SIFT [9], SURF [11] and BRISK [12]. Most key-point detectors have a number of desirable qualities such as scale invariance, illumination invariance, rotation invariance and robustness. However, each key-point that is detected requires an analysis of the surrounding region to create a feature descriptor. In the case of SIFT features, this is a 16×16 pixel region centred on the key-point. Object tracking with key-points is achieved by matching key-point descriptors of a set of points to all detected points in the next frame. Due to image noise, appearance changes and viewpoint changes, the points detected in each frame will not perfectly match and thus multiple points must be detected on the target in order to track it accurately. Considering the size of the sensors available in SPAD cameras, it is unlikely that a target will occupy a large enough region for these algorithms to have a suitable number of key-points with which to track the target. Based on this, we will not explore point tracking methods further.

Silhouette or contour methods provide a better way to represent complex and irregular objects. However, they require enough detail in the image in order to distinctly resolve the edges and corners of the target. Also, given the nature of the objects likely to be tracked - namely large, rigid objects - the additional complexity of creating a silhouette based model to represent the target is not justified. When sensor technology has improved, silhouette tracking methods may become more practicable and also assist in object identification.

3.2 Kernel Methods

Kernel methods for tracking were first popularised in 2003 by Comaniciu et. al. [6]. The concepts they develop have been incorporated into many of the current state-of-the-art trackers. The underlying foundation that forms the basis for kernel methods is the small motion assumption. Using the current known location of the target, a similar region in

the subsequent frame is then analysed to find the best matching area for the new target's location.

The algorithm developed by Comaniciu et. al. uses an RGB colour space quantised into bins as the feature space. The target, represented by an elliptical region, is described as a histogram of these RGB features. Each pixel's contribution to the histogram is weighted by the kernel function - an isotropic, monotonically decreasing function - applied to the distance of that pixel from the centre of the target. This has the effect of biasing the histogram towards pixels located near the centre of the target. To locate the target in subsequent frames, a mean-shift algorithm is used to iteratively move the target location until it is at the point which most closely matches the target from the previous frame.

While this method shows promising performance, it has a number of major deficiencies. Firstly, if the movement of the target is large enough so that it moves entirely out of the target ellipse in the previous frame, then the tracker will be unable to locate it. For small targets, this can become a serious issue. Additionally, once the target is lost, its location will never be recovered. This means that any occlusion will result in the target being lost for the remainder of the sequence. Another major issue that is present in this and many other tracking algorithms is that it is not robust to dramatic appearance changes between frames.

Yilmaz [13] proposes an extension to the mean-shift kernel method by using a contour based object model and including the target's rotation and scale as additional dimensions in the mean shift. This modified algorithm is better able to track irregularly shaped objects as it negates the issue of the background being included as part of the kernel. However, it still does not address the issue of target loss and occlusions.

3.3 Machine Learning

Another approach that has emerged more recently is to use machine learning to train the tracker, also known as adaptive tracking by detection. The common concept behind these trackers involves using the known target location to train a classifier to discriminate between the target and the background. This is then used to determine the target location in subsequent frames. The classifier is then updated based on the new target's location and features.

Avidan [14] trains a multitude of weak linear classifiers to classify a pixel as being part of the background or the target. Using the AdaBoost method [15], these weak classifiers

can be combined to produce a strong classifier. Using this classifier, a confidence map can be generated for the subsequent frame based on how confident the classifier is that each pixel forms part of the target. Rather than calculate this map for the entire image, the same mean-shift algorithm as above is used to locate the region of maximum confidence, and thus the new target location. The feature space used was an 11 dimensional space, which included the R,G,B values of the pixel, and "an 8-bin local histogram of oriented gradients calculated on a 5x5 window" [14]. Another novel aspect to this work is that the classifiers were re-trained over time. After finding the target in each frame, the worst performing classifiers are removed and new classifiers are trained on the current frame to replace them. In addition to this, the weightings of these classifiers decreased over time, favouring newer classifiers that had been trained on more recent views of the target. Because of this 'memory', the tracker is robust to short term occlusions.

One problem that is prevalent in these types of trackers is their susceptibility to drift. Small inaccuracies in the detection of the target compound over time, especially when the current target is being used to train the model. Babenko et. al. [16] propose a way to counter this problem by using Multiple Instance Learning (MIL). The concept behind the *MILTrack* algorithm [16] is to train a linear classifier on patches of the image, classifying them as either background or target. However, instead of simply using a single patch from the target, a set of patches within a small radius of the target's location are used. This accounts for the fact that the estimated location of the target may not be exactly correct and reduces the effect of drift. Similar to Avidan, a set of weak linear classifiers is used and updated on-line to account for appearance changes in the target. This approach achieves good results on a number of challenging video sequences and was one of the leading algorithms upon its release in 2009.

Building on the concept of multiple instance learning, the *STRUCK* algorithm [8] replaces the classifier with a structured-output Support Vector Machine (SVM). The key difference of this design is that the SVM is trained to output target transformations directly, rather than the traditional approach of using a classifier to identify potential targets and using a different method to determine the transformation. The SVM is trained online, and in each frame positive and negative support vectors can be added, with positive support vectors representing a particular appearance of the target. For example, when tracking a person's face, the positive support vectors would consist of image patches showing their face from different angles, rotations, illuminations and expressions. Over time, the SVM then learns the array of different appearances the target may have, making it more robust to rapid appearance changes and occlusions that other trackers would fail to accurately detect. This additional complexity comes at a significant performance cost and *STRUCK*

is one of the slower algorithms in the current state-of-the-art.

Henriques et. al. [17] take a significantly different approach to the problem with the Kernelised Correlation Filter (KCF). They argue that the main limiting factor in many tracking algorithms is the focus on positive target samples, with little emphasis on negative, or background, samples. By using a simple linear regression model and exploiting a property of circulant matrices in the Fourier domain, they are able to train a classifier on thousands of samples very efficiently. The training set is composed of circular shifts of the original target image patch, with the regression target being related to how much of a shift the sample has undergone. The set of these samples can be combined to create a circulant matrix which can be diagonalised by applying a Discrete Fourier Transform. A consequence of this diagonalisation is that the complexity of training the linear regression model on these samples becomes $O(n \log n)$ as opposed to $O(n^3)$. The algorithm is able to work with a number of different feature descriptors, including raw RGB values and also histogram of gradients. Detection of the target is performed simply by evaluating the linear regression model on the new frame in the region surrounding the previous target's location.

While the KCF tracker is one of the best performing trackers currently available, it still has several weaknesses. Compared to *STRUCK*, it does not keep a long term history of the target's appearance, and thus struggles to reacquire a target after an extended occlusion or out-of-view event. Similarly to the Meanshift algorithm, it is also susceptible to tracking the wrong object if a second candidate passes in front of the target. The authors admit that there is still a wide range of additions that can be made to address these issues, but the basic algorithm is still competitive against significantly more complex trackers.

4 Selection and Implementation of Trackers

In order to effectively assess the viability of visual object tracking algorithms on SPAD camera data, we selected a representative sample of trackers for comparison. The trackers selected were Meanshift [6], KCF [17] and STRUCK [8], as these represent the dominant state-of-the-art approaches taken towards visual object tracking. While it should be noted that there have been more recent developments in higher performant trackers [18], many of these build on the techniques and approaches that were developed in the selected trackers. One example of this is that the selected trackers do not provide a way to detect when the target has been lost. If there is any occlusion or the target moves out of the frame, then they also have no ability to re-capture the target when it becomes visible again. While

extensions to these trackers have been developed to incorporate these features, these are outside of the scope of this project. Thus, we do not expect the algorithms selected to provide the best possible results, but will rather provide an indication as to which family of algorithms is better suited to tracking objects using the SPAD camera data. Using this information, we can guide the research in this area towards algorithms that are more likely to succeed, and avoid expending effort on those that show little potential.

An evaluation suite was developed in MATLAB to perform the testing of each tracker[1]. Implementations for both Meanshift and KCF were created, based off the original authors' specifications and example code. These were tested and compared to the existing implementations to confirm that they matched the algorithms' stated performance. In the case of STRUCK, the original C++ code was used and a MATLAB wrapper was written to interface it to the evaluation tools. Due to the complexity of the tracker and also the performance requirements, it was decided that re-implementing the STRUCK tracker in MATLAB posed too great of a risk, both in terms of potential for errors and in performance losses.

Following the same approach as the VOT2016 challenge [18], the estimated target location from a tracker is monitored while it is running on a video sequence. If the target drifts too far from the ground truth then the tracker is re-initialised with the current ground-truth location. This compensates for the trackers inability to detect and correct for loss-of-target situations. It also reduces the bias that video sequence length has on the results. For example, if a tracker loses the target at the beginning of a long sequence, it can be reset and still accurately track the target for the remainder of the sequence. Without resetting, the tracker would fail to track any subsequent frames and its performance would heavily depend on what time the target was lost. The number of times that a tracker needs to be reset can be recorded for each sequences and used a measure of robustness, with a lower number of resets indicating better performance. Ultimately, this method is only useful as an evaluation tool as it requires a pre-existing ground-truth trajectory for the target, which is not available in real-world tracking scenarios.

5 Analysis on Benchmark Datasets

Before attempting to run the trackers on the SPAD camera data, we first performed an analysis of their performance on standard RGB video sequences. The purpose of this is to

[1] All code for this project is available at https://bitbucket.org/jackhenderson101/engn3712-code/

Name	Frames	Camera	Image Dimensions	Average Target Size	Source
Biker	142	Fixed	360×640	24×31	[19]
Box	1161	Fixed	480×640	96×122	[19]
Car1	1020	Moving	240×320	25×21	[19]
egtest01	1821	Moving	480×640	33×23	[20]
egtest02	1301	Moving	480×640	33×22	[20]
egtest03	2571	Moving	480×640	34×30	[20]
Panda	1000	Fixed	233×312	28×21	[19]
RedTeam	1918	Moving	240×352	37×20	[19]
Surfer	376	Moving	360×480	31×37	[19]

Table 1: Description of selected benchmark sequences

provide a performance benchmark using sequences that these algorithms would typically be run on. Sequences were selected from different sources [19], [20] focussing on those with low resolution, as this is one of the defining aspects of the SPAD camera data. Sequences with occlusion and target-out-of-frame characteristics were avoided as the primary focus was on simple tracking behaviour. A summary of each sequence is shown in Table 1.

5.1 Metrics

It can be difficult to objectively rate the performance of a tracker on a sequence of data. Ultimately, it depends on the intended application of the tracker, and what aspects of the tracker are more important for the particular application. Two of the most commonly used metrics are target location error, and target overlap [8], [17], [18], these provide two different but equally valuable measures of tracker performance. The target location error is simply the euclidean distance between the centre of the target and the centre of the ground truth. In many cases, pixels are used to measure the distance between the two points, however this creates a dependence on the resolution of the image, and the size of the target in the frame. In order to correct for this, the location error is normalised against the dimensions of the ground truth target:

$$\sqrt{\left(\frac{T_y - G_y}{G_h}\right)^2 + \left(\frac{T_x - G_x}{G_w}\right)^2} \tag{1}$$

where T is the target location determined by the tracker and G is the ground truth value. Here, the subscripts x and y denote the x-coordinate and y-coordinate of the centre of the

rectangle and the subscripts w and h denote the rectangle's width and height, respectively.

The target overlap metric measures the area of the intersection of the target and ground truth rectangles as a fraction of union of the areas. This allows the metric to incorporate information about how closely the shape and scale of the target matches the ground truth, rather than just its relative position. Similar to the normalised location error metric, this is invariant to the resolution of the image. However, as most of the datasets selected have small bounding boxes, there is unavoidable discretisation on the dimensions of the ground truth. This can mean an error of only one or two pixels can have a significant penalty on the value of the metric.

5.2 Parameters and Features

The three selected trackers were run on the datasets shown in Table 1. Parameters for each of the trackers were kept at the default values described in their respective papers [6], [8], [17], with a detailed description of these parameters listed in Appendix A. The features used in this trial were just the raw pixel values from the image. This provides a baseline indication of performance and allows comparisons to be made between the trackers without biasing the results to trackers that use better features. Also, because of the nature of the SPAD image data, it is likely that an entirely new feature will need to be created in order to ensure the best tracking performance.

5.3 Results

5.3.1 Surfer Sequence

We highlight the 'Surfer' sequence as a typical tracking sequence, with several key frames shown in Figure 1. As shown, the ground truth follows the surfer's head and all trackers perform moderately well. Figure 2 shows the normalised location error over time, from which several observations can be made. STRUCK performs consistently well with a high percentage of frames having an error below 0.4. It was also not reset in this sequence, indicating that the tracker correctly tracked the target across every frame. Compared to this, the Meanshift tracker was reset 4 times, or at a rate of 1.06 resets per 100 frames. It also tracked the target less-precisely than the other two trackers, as the location error is consistently higher. The KCF tracker was often able to track the target more precisely than STRUCK, for example between frames 60 to 110, and 250 to 370. However, there

Figure 1: Key frames selected from 'Surfer' sequence, showing the target identified by each tracker and the ground truth. (Best viewed in colour)

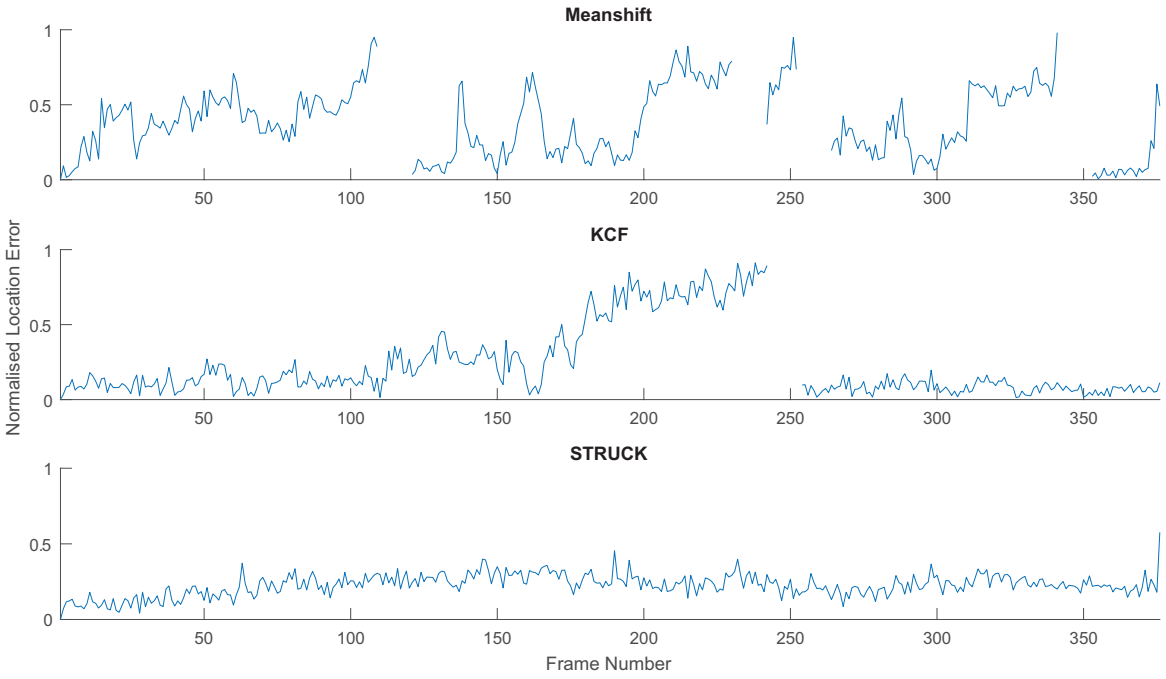
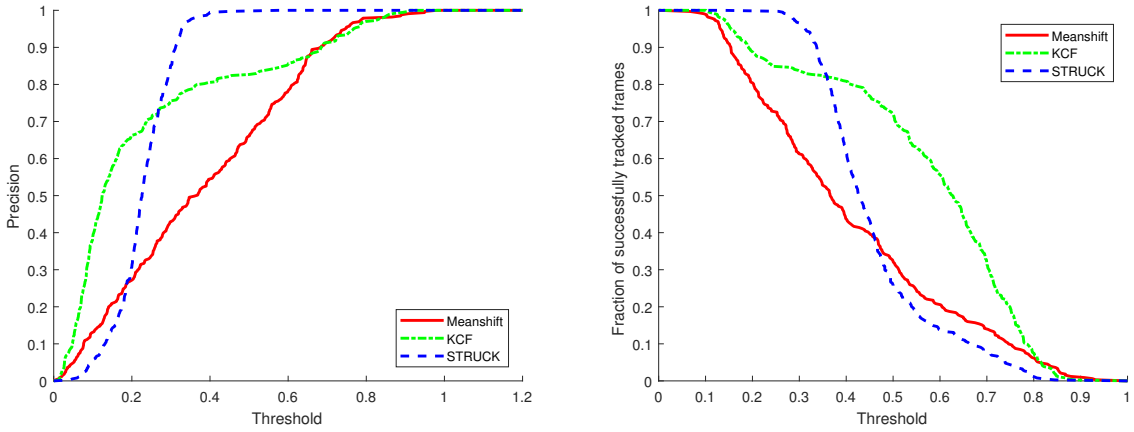


Figure 2: Normalised location error over time for 'Surfer' sequence.



(a) Precision Curve *(b) Success Curve*

Figure 3: Precision and Success curves for 'Surfer' sequence

was a section in-between where it drifted away from the ground truth and was eventually reset.

The performance of the trackers can also be evaluated using precision and success curves, see Figures 3(a) and 3(b) respectively. The precision curve indicates the fraction of the frames where the location error was below the specified threshold. As the threshold increases, more frames meet the criteria and thus the curve monotonically increases towards 1. Steeper curves and higher values for lower thresholds indicate better tracking performance. The success curve measures the fraction of frames where the target overlap is above the specified threshold. In this case, flatter curves and high values for high thresholds indicate better tracking performance. In both cases, the closer the curve is to a horizontal line with value 1, the better the performance.

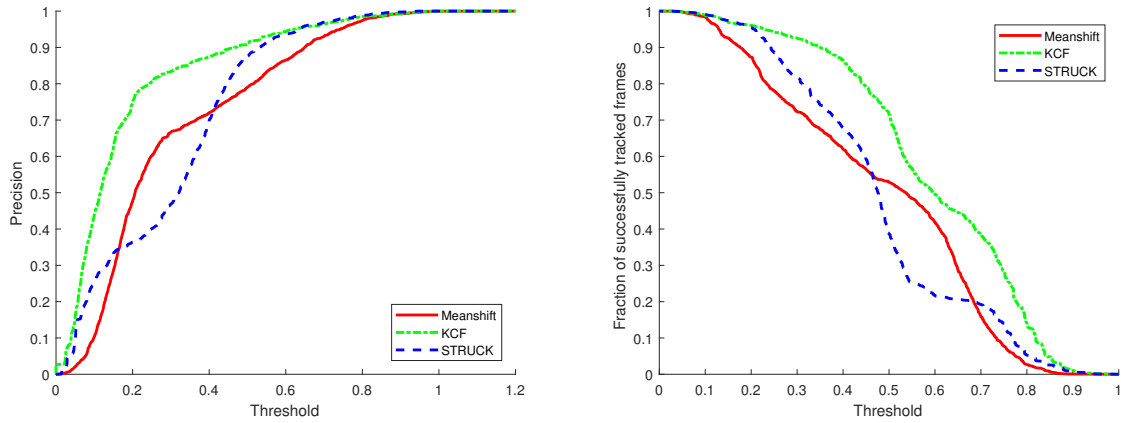
The precision and success curves for the 'Surfer' sequence, shown in Figure 3 reveal the same information that was previously discussed. The KCF tracker has a higher precision than STRUCK for approximately 70% of the sequence, but on the whole sequence STRUCK provides better performance as indicated by the precision curve approaching 1 at a much lower threshold than for KCF. The success curve reveals one of the limitations of the STRUCK tracker used. While it may track the location of the target accurately, the implementation that we used does not have scale adaptation as part of the algorithm. This means that the size of the bounding box used by the tracker is fixed, and cannot adjust when the target changes in size. This can be observed in the success curve by the poor performance at thresholds above 0.5 and visually in frame 150 of Figure 1. The STRUCK bounding box is wholly contained within the ground truth, but because of the incorrect scale, it is scored poorly on the target overlap metric. While scale adaptation

Figure 4: Key frames selected from 'egtest02' sequence, showing the target identified by each tracker and the ground truth. (Best viewed in colour)

was a feature that was added to STRUCK after the initial release [8], it was not included in the software that was publicly available.

5.3.2 Egtest02 Sequence

The 'egtest02' sequence, shown in Figure 4, provides an example of a more challenging scenario. The target is much smaller, changes size dramatically, and there are other vehicles in the scene that pass near the target vehicle. At first glance, the precision and success curves for this sequence, shown in Figure 5, indicate that the performance of KCF is significantly better than STRUCK or Meanshift. However, it should be noted that KCF and Meanshift were reset 10 times, and STRUCK was reset only 6 times. A notable example of where STRUCK outperforms KCF in this sequence is in frame 512, visible in Figure 4. Another vehicle passes behind the target travelling in the opposite direction. It has a higher contrast against the background compared to the desired target, and the KCF tracker mistakenly begins to track this vehicle. As the STRUCK tracker maintains a set of support vectors, describing the different appearances of the target, it is able to distinguish between the two vehicles, despite the bounding box encompassing both. STRUCK again suffers heavily in this sequence due to the failure to adapt to the decreasing target size.



(a) Precision Curve *(b) Success Curve*

Figure 5: Precision and Success curves for 'egtest02' sequence

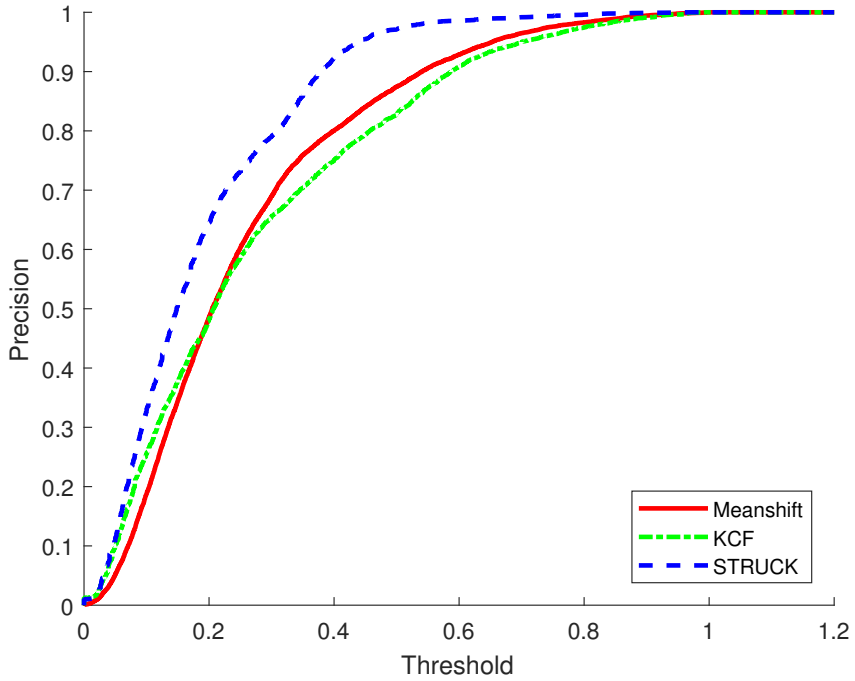


Figure 6: Combined precision plot for all benchmark sequences

	Successfully Tracked Frames (LE<0.25)	Successfully Tracked Frames (TO>0.4)	Resets per 100 frames
Meanshift	60.3%	55.9%	0.42
KCF	58.4%	64.1%	0.25
STRUCK	73.1%	69.3%	0.11

Table 2: Summary of results from benchmark sequences

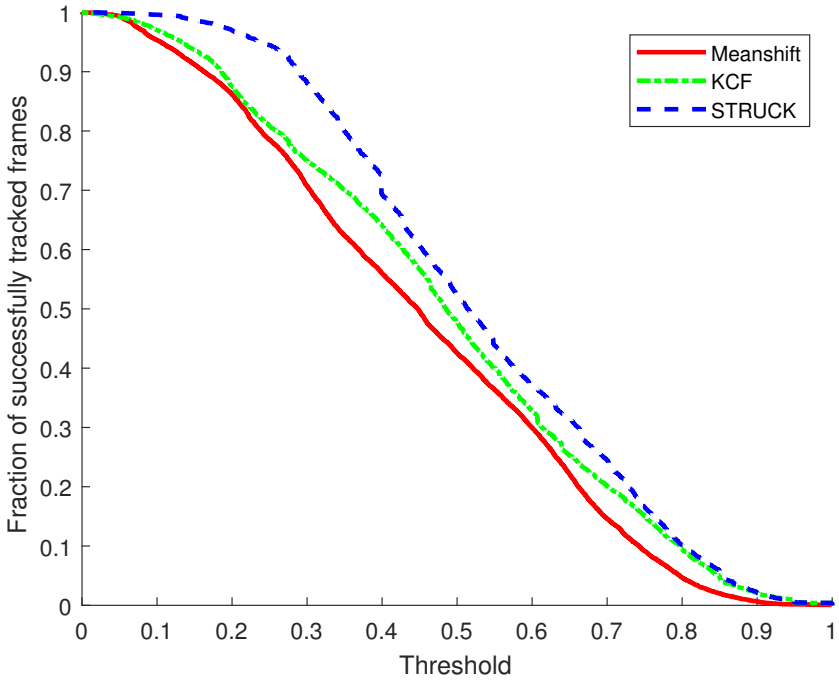


Figure 7: Combined success plot for all benchmark sequences

5.3.3 Overall Analysis

Analysing the set of sequences as a whole provides a different, and somewhat surprising perspective. The precision and success curves shown in Figures 6 and 7 respectively, clearly show that STRUCK performs the best out of the three. However, the distinction between KCF and Meanshift is not as clear. KCF tracks a larger percentage of the frames very accurately, as indicated by the high precision score at small thresholds, but falls short of Meanshift as the threshold is increased. This suggests that KCF is a more accurate tracker but fails more abruptly than Meanshift does. On the contrary, the success curves of both trackers show that KCF provides consistently better tracking than Meanshift, which may indicate that the deficiencies in tracking accuracy mentioned above are compensated for by better target scale adaptation. Interestingly, STRUCK again performs well on this metric even though it lacks the scale adaptation of the two other trackers.

Table 2 shows a summary of the algorithms' performance based on a single threshold value for the Location Error (LE) and the Target Overlap (TO) metrics. The reset rate per 100 frames gives an indication of the robustness of the tracker and shows that STRUCK performs best against this metric as well. As the reset rate of Meanshift is nearly double that of KCF, it will inherently have an advantage in terms of tracking accuracy. This suggests that despite the two trackers having similar precision curves, KCF is much more

robust and without target resetting it would significantly outperform Meanshift.

5.4 Validation

The results obtained from these trials are broadly consistent with those discussed in the literature. Meanshift provides moderately good performance, but cannot match the newer and more advanced trackers like STRUCK and KCF. The literature is not as clear on the differences between STRUCK and KCF. As is typical, the authors of these algorithms claim that they can achieve better performance than the alternative [8], [17], but independent benchmarks [18] and the general consensus indicate that KCF is more performant than STRUCK. However, this is based on the best performing set of features for each tracker, rather than just the raw pixel values. Henriques et. al [17], authors of KCF, show that STRUCK does indeed perform better than KCF with raw pixel features. It also performs better than all feature variants of KCF on the 4 sequences they tested designated as low resolution.

It appears that the key feature of KCF that allows for its high performance is also one of its weaknesses. The utilisation of circulant matrices and Fourier transforms allows KCF to train on a much larger number of samples than other algorithms. However, as the size of the target decreases, this advantage diminishes and the trade-offs that have had to be made to achieve this become evident. STRUCK trains with much fewer samples, by taking a random selection in the neighbourhood of the target, but performs a much deeper analysis of each sample rather than just relying on the sheer volume of samples in the way that KCF does.

6 SPAD Data Modelling

A limited set of SPAD sensor data was provided by the Defence Science and Technology Group (DSTG) for use in this project which contained data from 3 different scenarios, each with 55 recordings. In each scenario, a micro-UAV was positioned in front of the sensor and moved between two fixed points in a specified amount of time. This was repeated 55 times for each scenario, to generate a total of 165 sequences. A qualitative description of the trajectory was provided for each scenario, however a true, independent ground truth measurement for each sequence could not be obtained due to limitations of the experimental setup. Without a ground truth accompanying the sequences, the initialisation, target

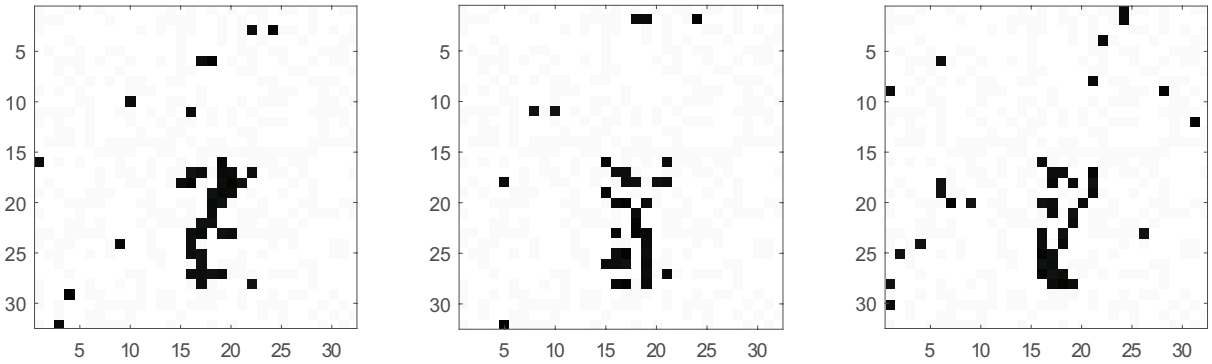


Figure 8: An example of 3 consecutive frames from the DSTG dataset. (Frame: 1634,1635,1636, Seq: MovingAwayDark1)

resetting, and most importantly the evaluation of the tracking algorithms could not be performed. Additionally, a hardware fault was identified in the camera, which suggested that the data received might have slightly different characteristics to true SPAD data. DSTG indicated that this fault would not significantly impact the overall properties of the data.

In order to account for a lack of ground truth, a synthetic dataset was created. Using the data provided from DSTG, the characteristics of the SPAD sensor data were studied and then replicated to generate artificial data that matched the real-world data as closely as possible. The advantage of this approach is that the sequences can be generated directly from a ground truth specification, and thus allow the algorithms to be evaluated against the true target trajectory. It also allowed for arbitrary trajectories to be evaluated and modifications of the target size and sensor size to be made.

6.1 SPAD Data Observations

The sensor used to collect the DSTG data was a 32×32 pixel array. The values measured by the sensor are correlated with the time delay of a photon incident to that pixel, and thus are correlated with the depth of the object in that region of the scene. There are inherent limitations to the resolution of the depth measurements as they were recorded with 10 bits per pixel. Moreover, the values are based on the precise in measurement of the time of arrival of individual photons. In the experiments that were performed to create the dataset, the target's depth changed by approximately 10 cm, which is close to the lower limit of what the sensor can resolve. Thus, the data available only contains measurements in a small range of values, and hence we cannot draw any conclusions about how these measurements are affected by target distance.

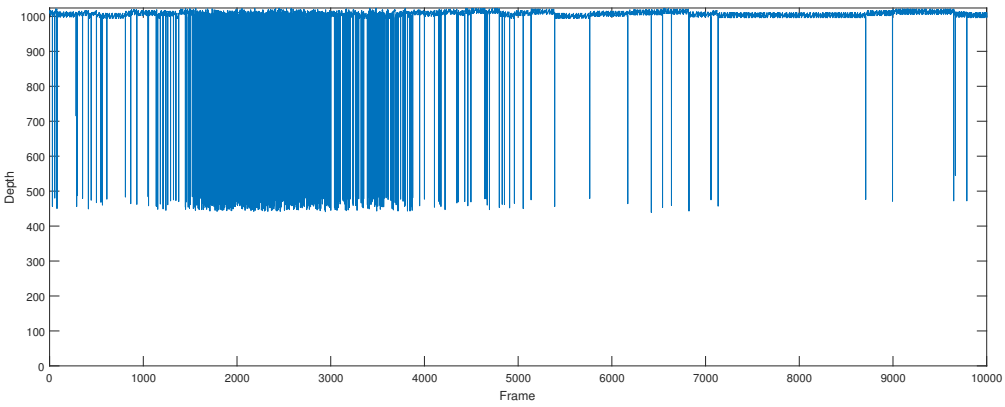


Figure 9: Measurements from a single pixel in a sequence. (Pixel $(20, 20)$*, Seq: MovingAwayDark1)*

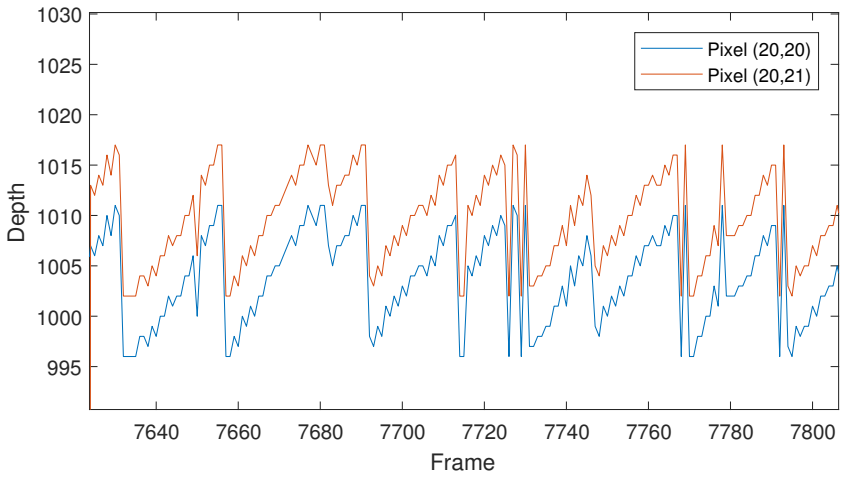


Figure 10: Measurements from two neighbouring pixels showing sawtooth phenomenon. (Seq: MovingAwayDark1)

Observation of the DSTG dataset indicated that there are two distinct phenomena; the low-amplitude background noise and the discrete high-amplitude pulses which form the signal. Figure 9 shows the behaviour of a single pixel in the SPAD array. The background noise is visible in the range $[990, 1023]$ and appears to be both spatially and temporally correlated. The 'sawtooth' behaviour of this noise can be observed in Figure 10 and is most likely an artefact of the sensor electronics. This observation was confirmed with DSTG. When compared to the overall signal, this noise has a relatively insignificant magnitude and appears to be independent of the other elements of the signal. It's likely that pre-processing would remove this noise before tracking algorithms were applied, and thus we do not consider the effect of this noise in our model.

The primary component of the signal is a set of discrete pulses that occur in the range of $[400, 600]$. It is believed that these pulses correspond to individual photons incident to that pixel. As seen in Figure 9, the frequency of these peaks changes over time, as does their value. The difference in peak value is caused by variation in the arrival time of photons to the sensor while the change in the frequency of pulses is due to the target moving across the frame. The frequency of the pulses increases as the target moves into the region that is observed by this pixel. There is also a significant amount of erroneous pulses when the target is not observed by the pixel, e.g. Frame 4000 onwards. These may be caused by stray photons from unintended reflections or other light sources. These phenomena can be observed in the spatial dimensions in Figure 8, which shows 3 single frames of the data. The target is approximately in the lower centre of the frame, but there are a number pulses outside the target region. This makes it difficult to exactly determine the target location using just a single frame of data.

6.2 Creating Synthetic SPAD Sequences

To create synthetic sequences, we model the data with two distributions for each pixel; the value of each pulse and the number of frames between consecutive pulses. In order to do this, we make two main assumptions. The first is that the distributions are independent of time, which only holds if the target does not move in the frame. The second assumption is that samples from adjacent pixels are independent of each other. This is a much more complex condition to verify as there is an overall correlation between adjacent pixels because they are measuring adjacent regions in space. If the target is observed by a particular pixel, then it's likely that the target will also be observed by adjacent pixels. However, in terms of the value of the pulses measured and the interval between consecutive pulses, the assumption is that these are spatially independent. Initial observations of the

data suggest that this is true, however a more rigorous analysis is required to confirm this assumption. One possible approach would be to use the Hilbert-Schmidt Independence Criterion (HSIC) [21] as a measure of whether the samples are independent.

Another assumption that we make is that the background noise is independent of the target location. This is not particularly realistic as, for example, photons from the target will be reflected by background objects, and this depends on the presence of the target in a particular location. Further to this, we also assume that the appearance of the target remains constant throughout the sequence. In practice, as the target moves in the frame there may be out-of-plane rotations, which will change the observed characteristics of the target. Both of these assumptions are necessary as they remove a significant level of complexity into the model and the DSTG dataset is not of high enough quality to facilitate modelling these factors.

Initially, the plan was to use an additional dataset from the SPAD sensor in which the target is stationary in the frame for the entire sequence. Using this, we would be able to determine the distributions for pixels that measure the background and those that measure the foreground (region containing the target). However, due to technical issues with the SPAD sensor that were outside of our control, this dataset was not made available within the time frame required by this project. Additionally, it was discovered that the original sequences received may also contain invalid data due to a fault in the sensor. Based on the advice that the fault was only minor, and that the properties of the data would be broadly similar, we continued with the analysis using the original data. However, any conclusions drawn from using this SPAD data must be conditioned on the fact the true SPAD sensor data may have slightly different properties.

As a sequence with a stationary target was not available, a subsequence of data from the existing dataset was identified where the target was stationary for several hundred frames. Frames 1 to 1000 of the "movingAwayDark01" sequence were selected and the location of the target was identified as $(T_x, T_y, T_w, T_h) = (24, 14, 14, 10)$. Using this subsequence, the distributions of peak value and consecutive peak times were evaluated for each pixel in the target region. Pixels not in the target region were considered background, and a single pair of distributions for all background pixels was evaluated.

The synthetic sequences were created by sampling from these distributions using the method of inverse transform sampling. Firstly, a sequence of frames was generated containing only background pixels. For each pixel in the frame, a set of i.i.d. samples were drawn from the value distribution and the consecutive pulse distribution. Using these samples, the pulses were added into the sequence at the appropriate times. Independent

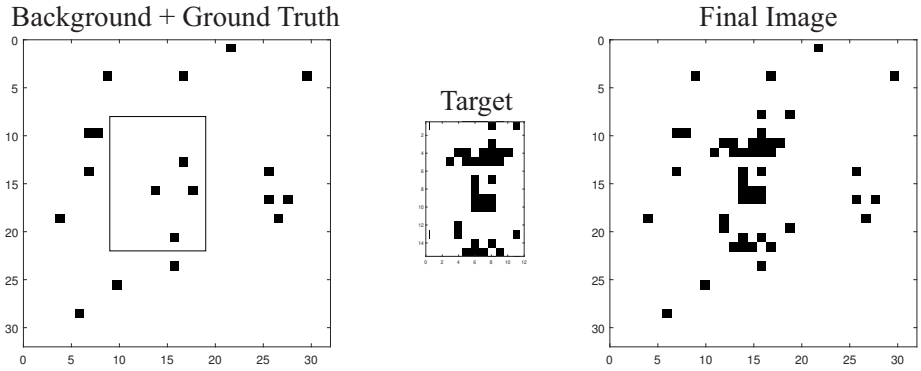


Figure 11: Graphical representation of the creation of a synthetic frame

of this, a target sequence was created with dimensions equal to the original target's width and height. Using the individual distributions for each pixel, the sequence of target frames was created in a similar way to the background. The target sequence was then overlaid onto the background in the location specified by the ground truth trajectory. Figure 11 provides a graphical description of this process.

7 Analysis on SPAD Datasets

As a model of the SPAD sensor data had been created, we were now able to create arbitrary datasets with a range of different trajectories and their accompanying ground truth bounding boxes. Five different sequences were created, each 1000 frames long and with a different trajectory. The specification of each of these sequences is provided in Table 3. A mix of simple linear and circular trajectories were selected to ensure that the trackers encountered a range of different target behaviours. The size of the frame was expanded from the original resolution of 32×32 to 50×50, while keeping the size of the target the same and its location within the original 32×32 region. This was to prevent the trackers from hitting the edges of the frame when the target is close to the boundary. If the size of the frame were not increased, trackers that reach the edge will be constrained to stay within the frame. This may result in better tracking accuracy than it would otherwise be observed if the target location wasn't constrained. As the target is modelled from an existing SPAD sequence, it retains the dimensions of that sample. In this case, the target's width and height are 10 and 14 pixels respectively.

Seq.	Initial Location	Final Location	Trajectory	Description
1	$(15, 15)$	$(35, 35)$	Linear	Top Left to Lower Right
2	$(15, 25)$	$(35, 25)$	Linear	Vertical through centre
3	$(25, 15)$	$(25, 35)$	Linear	Horizontal through centre
4	$(40, 25)$	$(40, 25)$	Circular	Anti-clockwise around centre
5	$(40, 25)$	$(40, 25)$	Circular	Clockwise around centre

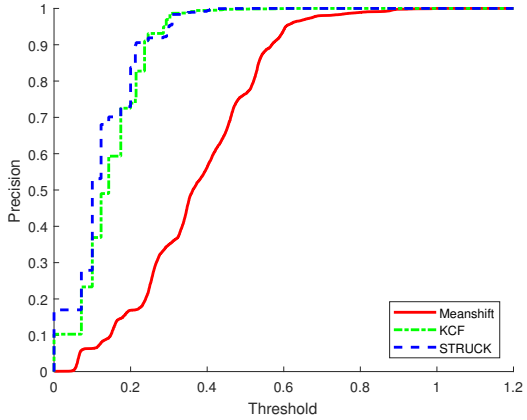
Table 3: Specification of synthetic SPAD datasets created.

	Meanshift	KCF	STRUCK	BlobTrack
Raw	23.52	93.10	91.90	—
Raw - 3x scale	93.48	91.90	98.52	—
Peak Count - 5 frames	88.79	99.94	100	29.53
Peak Count - 10 frames	97.86	77.19	89.69	59.10
Peak Count - 30 frames	100	34.20	78.20	100

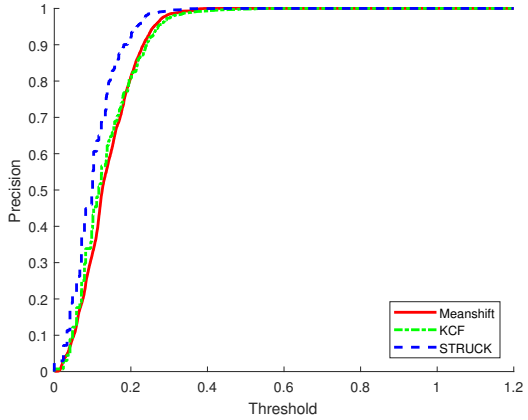
Table 4: Percentage of successfully tracked frames where Location Error < 0.25

	Meanshift	KCF	STRUCK	BlobTrack
Raw	50.72	99.18	99.24	—
Raw - 3x scale	99.88	99.20	99.94	—
Peak Count - 5 frames	100	100	100	44.00
Peak Count - 10 frames	100	92.53	99.74	87.20
Peak Count - 30 frames	100	55.95	86.92	100

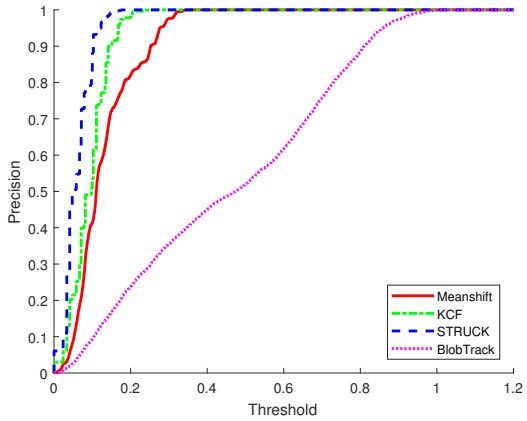
Table 5: Percentage of successfully tracked frames where Target Overlap > 0.4



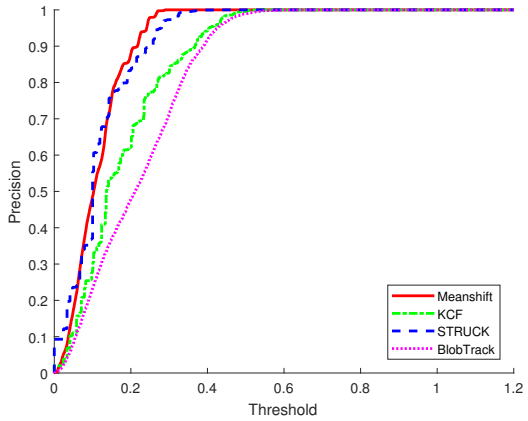
(a) Raw feature



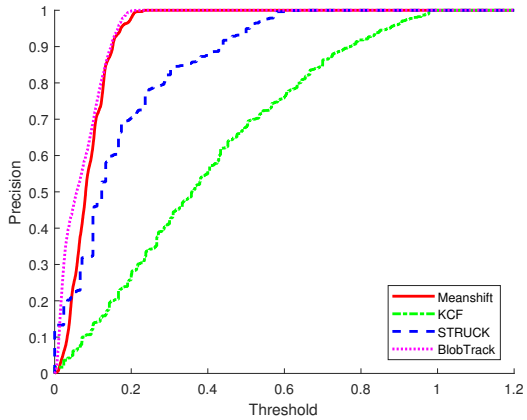
(b) Raw feature with 3× scale



(c) Peak-count feature over 5 frames

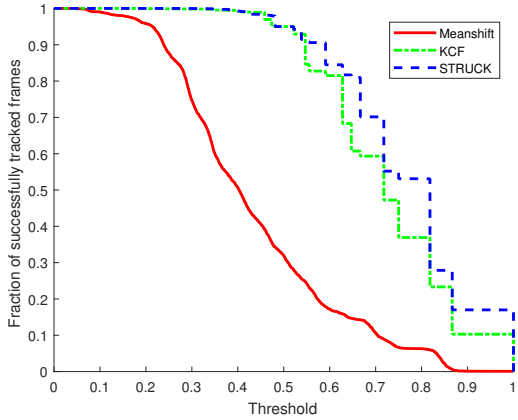


(d) Peak-count feature over 10 frames

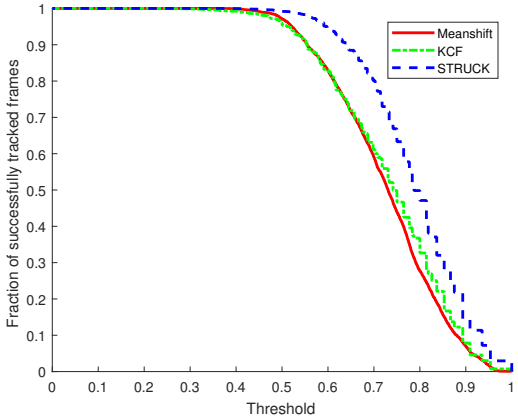


(e) Peak-count feature over 30 frames

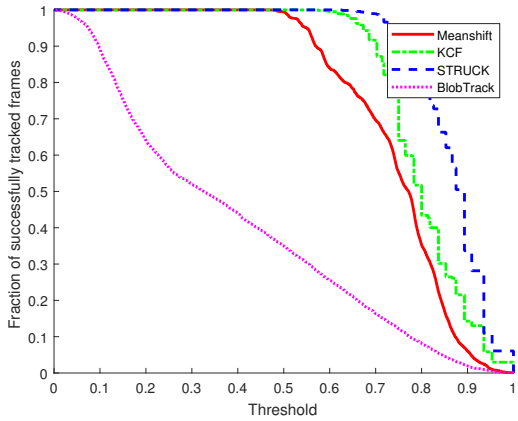
Figure 12: Precision curves for total set of synthetic SPAD sequences



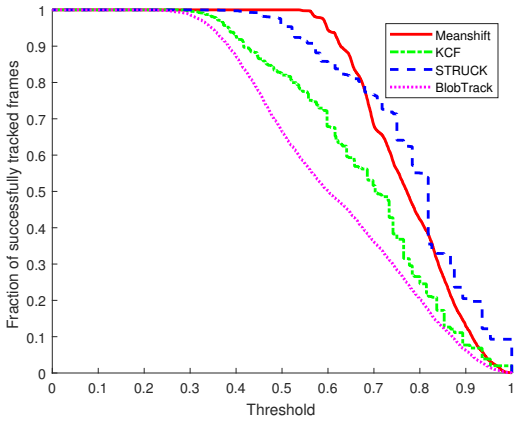
(a) Raw feature



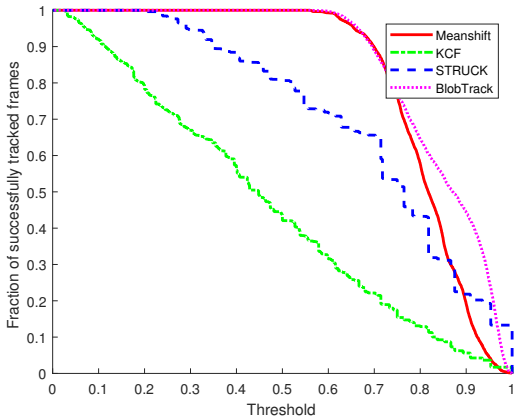
(b) Raw feature with 3× scale



(c) Peak-count feature over 5 frames



(d) Peak-count feature over 10 frames



(e) Peak-count feature over 30 frames

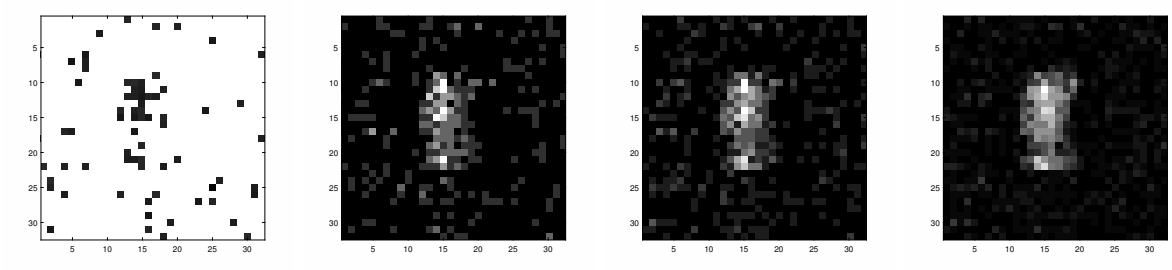
Figure 13: Success curves for total set of synthetic SPAD sequences

7.1 Tracking with raw data

The first test performed was to run the trackers on the raw, unmodified SPAD data. Precision and success curves are shown in figures 12(a) and 13(a) respectively. Surprisingly, all trackers performed moderately well on the raw data. STRUCK and KCF performed similarly, with Meanshift unable to match their performance. Initial testing showed that the significantly lower resolution of the target may be having an effect on tracking performance. Thus, another test was run with the SPAD image being scaled up by a factor of 3. This improved the performance of all trackers, as can be seen in figures 12(b) and 13(b). Meanshift improved the most, bringing it to a similar performance level as KCF, which itself did not benefit significantly from the scale change. Despite the high noise levels that are not typically found in traditional visual object tracking sequences, all trackers can accurately and reliably tracking the synthetic SPAD target throughout the entire sequence. STRUCK has a slight edge over KCF and Meanshift, and with the 3× scale change, tracks 98.5% of frames with a location error of less than 0.25. In to the benchmark sequences tested previously, STRUCK only successfully tracked 73.1% of frames according to the same criterion. This suggests that any of these trackers show potential towards successful tracking on real-world SPAD sensor data.

7.2 Tracking with peak-count feature

While tracking on the raw SPAD data proved successful, there is still opportunity for improvement. The more robust and accurate a tracker can be made on the synthetic data, the better chance it has when tracking more realistic data. Creating a feature that is invariant to the noise present in the SPAD data would provide a more stable input to the trackers, allowing them to track the target more accurately. Observing that the frequency of pulses in the signal is highly correlated with the location of the target, we propose a simple 'peak-count' feature. For each pixel, we detect peaks by observing when the value is below 950, which is sufficient to eliminate the background noise. We then count the number of these peaks in the last n frames of the sequence and use this as the feature value. We also retain the 3× scale factor used with the raw features. A larger choice for n will result in a more time-invariant feature, however it will lag behind the true target location, as it is relying on frames further in the past. If the target velocity is small then this effect is diminished. We test this feature with three different values of n – 5, 10 and 30 – to examine what effect this has on tracking performance. An example frame for each of these features is shown in figure 14.



(a) Raw feature *(b) $n = 5$* *(c) $n = 10$* *(d) $n = 30$*

Figure 14: Example of peak-count feature

Initial observations of the peak-count feature suggest that even a simple tracking-by-detection approach could be used to track the target. Considering this, we implement a simple tracker which determines the target location by computing the spatially-weighted average of the peak-count feature. In other words, it computes the centre of mass of the image. While this type of tracker has a vast number of potential drawbacks, it can be used as a baseline performance indicator to compare with the other trackers. If a given tracker fails to outperform this rudimentary algorithm, then it is clearly below the acceptable standard. We name this tracker 'BlobTrack' and include its results alongside the other three other trackers being tested to provide a baseline comparison.

Tracking results for the peak-count feature are shown in figures 12 and 13, as well as in tables 4 and 5. Interestingly, this feature affects each tracker differently. Starting at $n = 5$, we observe that Meanshift actually performs worse than the raw feature at 3x scale with successfully tracked frames dropping from 93.5% to 88.8%. However, increasing n to 10 and further to 30 yields significant benefits for Meanshift. Successfully tracked frames increases to 97.9% for $n = 10$, and 100% for $n = 30$. This clearly indicates that the peak-count feature provides a clear and significant performance improvement to the Meanshift tracker. In the synthetic data used, target movement was small enough to have a negligible lag effect on the target location.

The results for KCF and STRUCK are surprisingly different to what was expected. For $n = 5$, the peak-count feature provides a small, but significant improvement both trackers' performance. For KCF, the number of successfully tracked frames increases from 91.9% to 99.9% when compared to the raw feature, and STRUCK increases from 98.5% to 100% by the same measure. However, beyond $n = 5$, the performance of both trackers decreases dramatically. At $n = 30$, KCF only successfully tracks 34.2% of frames, which is the lowest score of any tracker, excluding BlobTrack, for any feature type. Performance for STRUCK is also degraded but not as severely, with successfully tracked frames dropping to 78.2%.

While these results are contrary to initial expectations, we theorise that the difference is due to the learning aspect of the KCF and STRUCK compared to Meanshift. In STRUCK and KCF, a model of the target is learned over time, and this is used to locate the target in subsequent frames. The more information that is used to train these models, the better performance will be. In the case of the peak-count feature, we predict that using a small value for n is useful to filter out background noise. However, increasing this value too much causes the feature to average out and discard data that is useful in training the models. Thus, we can observe that performance for both STRUCK and KCF diminishes as n increases as they are being trained on less informative data. For the more simple trackers, Meanshift and BlobTrack, performance increases as the target is more clearly distinguished from the background.

Overall, we can conclusively say that STRUCK performs the best out of the three trackers, with KCF lagging slightly behind. The best feature for these two trackers was the peak-count feature with $n = 5$. Meanshift performed best when using the peak-count feature with $n = 30$, but lagged behind the other trackers in most cases. The results show that the addition of the peak-count feature can improve tracking performance when compared to the raw feature, however this is heavily dependent on the number of frames included in the peak counting. Robustness does also not appear to be an issue, as there were no target resets in the entire sequence.

While there are a number of limitations and simplifications inherent to using the synthetic SPAD data, the results obtained have been valuable in a number of ways. Firstly, we can confirm that all trackers are able to accurately and robustly track a target with SPAD data. We have also identified the best performing tracker as well as the most suitable feature to use for each tracker. We discover that a simple tracking-by-detection approach can achieve comparable results to even the best tracker. All of these results suggest that all three trackers have the potential to perform well on more challenging datasets, and potentially on real-world SPAD sequences.

7.3 Tracking on DSTG Dataset

As we have established that all three trackers are more than capable of tracking the target on the synthetic data, we can provide a brief analysis on their performance on the original DSTG dataset. As discussed previously, without a ground truth reference for the dataset, a quantitative analysis cannot be performed. However, as we can observe in figures 12(e) and 13(e), BlobTrack comes very close to tracking the target near-perfectly.

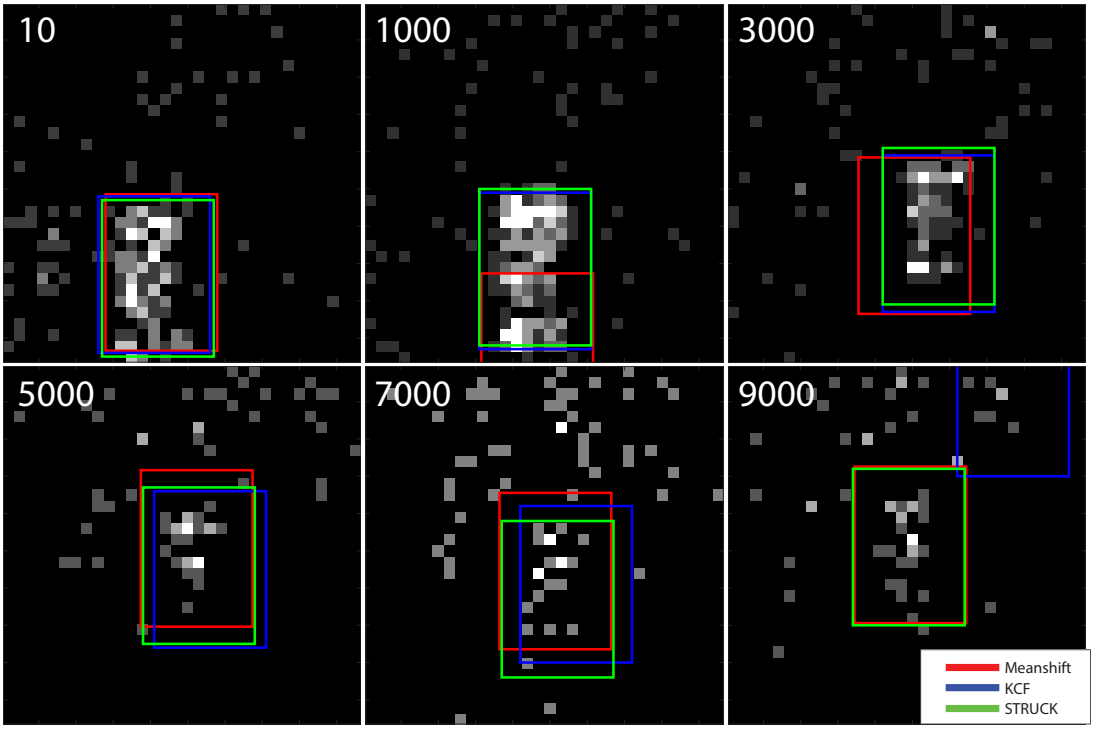


Figure 15: Key frames selected from 'MovingAwayDark1' sequence with tracking results. Frames visualised with peak-count feature ($n = 5$). (Best viewed in colour)

We can therefore use BlobTrack as a pseudo-ground truth reference in order to evaluate the metrics. There are inherent dangers in trusting BlobTrack as the ground truth, and thus we will not present any quantitative results such as the percentage of successfully tracked frames. In addition to this, we do not perform target resetting on the trackers as we cannot accurately reinitialise the trackers with the correct target location. We also cannot report on the target overlap metric, as the target changes size in the sequence and BlobTrack only tracks a single point, rather than a bounding box.

Figures 15, 16 and 17 show the raw results of running the trackers on a real SPAD dataset, while using BlobTrack as the ground truth. Beyond frame 5000, the target size begins to decrease, which means the ground truth values generated from BlobTrack are more sensitive to the noise in the signal, and thus the target location is not as consistent. This can be observed in the second half of the sequence, where the variance in error increase significantly, shown in figure 16. Overall, it shows impressive performance from all three trackers, which generally are able to track the target through the whole sequence. The only exception is KCF, which appears to lose the target twice, at frames 7685 and 9952. However, in the first case, it appears to re-capture the target at frame 9444 without any external resetting taking place, although this is most likely a coincidence rather than a true re-capturing. This long period of target loss for KCF explains the plateau

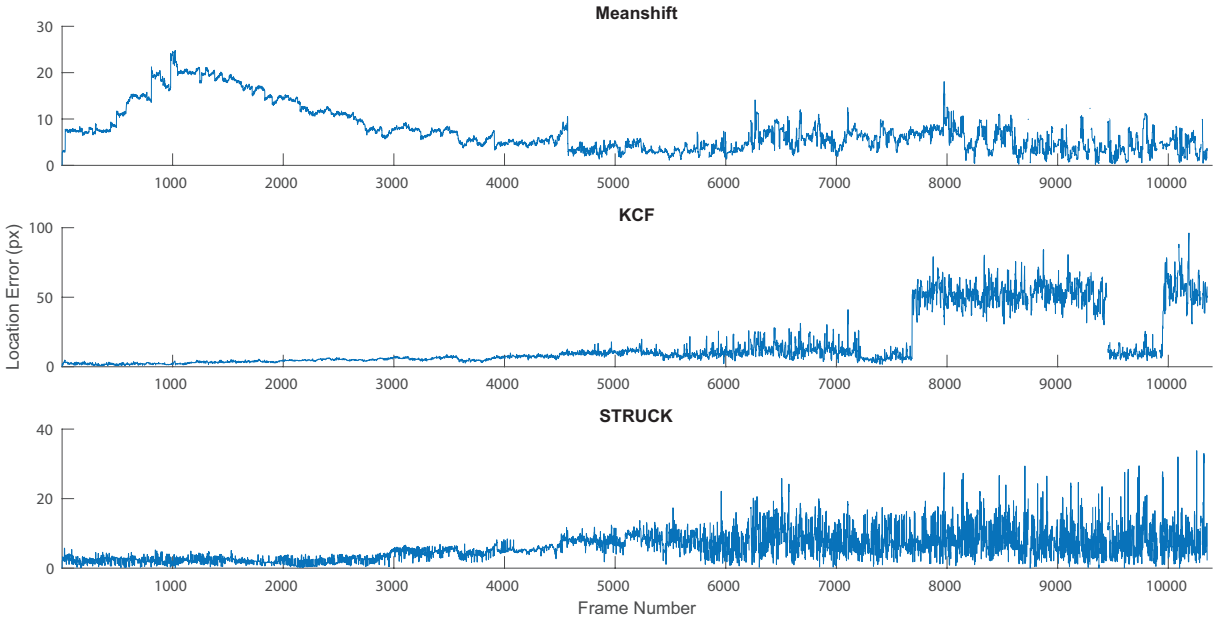
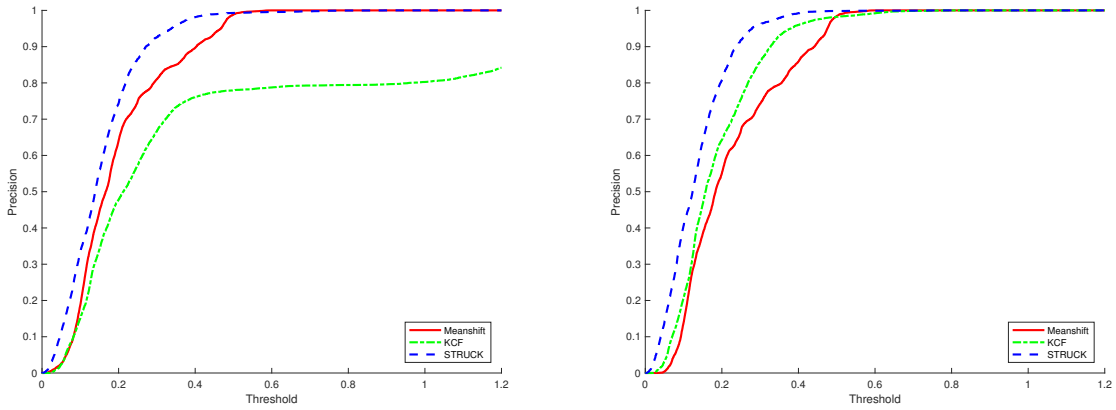


Figure 16: Non-normalised Location Error of each tracker over time on the MovingAway-Dark1 sequence, using BlobTrack as the ground truth.



(a) All frames *(b) Up to frame 7500*

Figure 17: Non-normalised precision curves for MovingAwayDark1 sequence.

observed in figure 17(a). Comparing the trackers only up to frame 7500 yields a more indicative comparison of performance, as shown in figure 17(b). However, without a reliable ground truth, the conclusions we can draw from this experiment are limited to general observations.

8 Conclusion

We examine the potential application of three visual object tracking algorithms on SPAD sensor data. Our research shows that the three algorithms selected, KCF, STRUCK, and Meanshift, represent the key foundational approaches that are present in most state-of-the-art trackers today. Evaluation of these trackers on standard benchmark sequences provides us with a baseline level of expected performance and provides some insight into the properties of each tracker. From these results, we observe that STRUCK performs particularly well on low resolution targets and is much more robust than KCF or Meanshift. While it is a significantly older and more primitive tracker, Meanshift still performs comparably to KCF, although suffers in robustness.

Our statistical analysis of the SPAD data revealed a number of key properties that were useful in the creation of a new image feature. In addition to information being encoded in the values of each pixel, there is also information in the frequency of pulses, which we utilise to create a new peak-count feature. We demonstrated that this feature can improve tracking performance compared with raw image data. Increasing the time-span of the feature show performance benefits for Meanshift, and the newly created BlobTrack, although due to the statistical learning nature of KCF and STRUCK, these trackers work best with smaller time spans.

The final test on a real SPAD sequence shows that the results from the model generalise well to real-world data. However, without an accurate ground truth trajectory, comprehensive analysis cannot be performed, and only qualitative results can be provided.

Overall, our results demonstrate that visual object tracking algorithms can be used to successfully track targets in SPAD sequences. Their performance and robustness were well above expected levels, which engenders positivity about the future potential of research in this area. The STRUCK algorithm demonstrated superior performance both in terms of accuracy and robustness and would be the best choice for further development.

9 Future Work

As visual object tracking with SPAD sensor data has been relatively unexplored previous to this, a number of potential avenues for further research still remain. The immediate next steps that follow from this research include developing additional image features and testing the trackers on more diverse and realistic datasets. In this project, we only investigated temporal aggregation of pulses, however there is potentially benefit in using spatial correlation of pulses in neighbouring pixels as well as the value of each pulse. Additionally, creating a high-quality dataset with range of different target trajectories, orientations, and appearances will prove invaluable to developing a feature that is invariant over more conditions and also better test the tracking capabilites of each algorithm.

One of the biggest limitations that all the trackers tested has is that they do not retain a model of the target's trajectory over time. The only way in which this is considered is through the small-motion assumption, which allows the trackers to limit the search space to a region around the previous location. The effect of this can be observed in the tracking results; the bounding box continuously jumps haphazardly from frame to frame as the algorithm attempts to find the exact location that minimises a given cost function. The underlying motion of the target however, is smooth, and so we observe a large amount of high-frequency noise in the error metrics. If the tracker also incorporates a dynamic motion model of the target, then this can also be used to augment the visual tracking. This would provide for a smoother target trajectory, and reduce the effect of noise in the image. This would be especially beneficial for use in SPAD sequences, as the signal-to-noise ratio is significantly lower than standard. A dynamical model would assist in the robustness to this noise and better predict the short to medium-term trajectory of the target.

References

[1] S. Cova, M. Ghioni, A. Lacaita, C. Samori, and F. Zappa, "Avalanche photodiodes and quenching circuits for single-photon detection," *Applied Optics*, 1996.

[2] M. Fishburn, "Fundamentals of cmos single-photon avalanche diodes," PhD thesis, Massachusetts Institute of Technology, 2012.

[3] C. Niclass, A. Rochas, P.-A. Besse, and E. Charbon, "Design and characterization of a cmos 3-d image sensor based on single photon avalanche diodes," *IEEE Journal of Solid-State Circuits*, 2005.

[4] A. Rochas, M. Gösch, A. Serov, P. A. Besse, R. S. Popovic, T. Lasser, and R. Rigler, "First fully integrated 2-d array of single-photon detectors in standard cmos technology," *IEEE Photonics Technology Letters*, 2003.

[5] S. Burri, Y. Maruyama, X. Machalet, F. Regazzoni, C. Bruschini, and E. Charbon, "Architecture and applications of a high resolution gated spad image sensor," *Optics Express*, 2014.

[6] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2003.

[7] J. M. Rehg and T. Kanade, "Model-based tracking of self-occluding articulated objects," *Fifth International Conference on Computer Vision*, 1995.

[8] S. Hare, S. Golodetz, A. Saffari, V. Vineet, and M.-M. Chen, "Struck: structured output tracking with kernels," *IEEE Transactions on Pattern Analysis and Machine Analysis*, 2016.

[9] D. G. Lowe, "Object recognition from local scale-invariant features," *7th IEEE International Conference on Computer Vision*, 1999.

[10] I. Haritaoglu, D. Harwood, and L. Davis, "W4: real-time surveillance of people and their activities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

[11] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: speeded up robust features," *Computer vision–ECCV 2006*, pp. 404–417, 2006.

[12] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: binary robust invariant scalable keypoints," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, IEEE, 2011, pp. 2548–2555.

[13] A. Yilmaz, "Object tracking by asymmetric kernel mean shift with automatic scale and orientation selection," *Computer Vision and Pattern Recognition*, 2007.

[14] S. Avidan, "Ensemble tracking," *IEEE Trans on Pattern Analysis and Machine Intelligence*, 2007.

[15] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, 1997.

[16] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," *Conference on Computer Vision and Pattern Recognition*, 2009.

[17] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[18] M. Kristan, A. Leonardis, J. Matas, and et.al, "The visual object tracking vot2016 challenge results," in *14th European Conference on Computer Vision*, 2016.

[19] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: a benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[20] R. T. Collins, X. Zhou, and S. K. Teh, "An open source tracking testbed and evaluation web site," *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2005.

[21] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf, "Measuring statistical dependence with hilbert-schmidt norms," in *Algorithmic Learning Theory: 16th International Conference*. Springer, 2005, pp. 63–77.

Appendix A Tracking Algorithm Parameters

Tracking on Benchmark Datasets

Meanshift

Image Feature	Raw RGB
Quantisation Bins	16 per channel
Max Iterations	20
Adaptation Rate, γ	0.05
Scale Adaptation	True
Scales Tested	$[0.9, 1, 1.1]$
Scale Weights	$[1, 1.05, 1]$

KCF

Image Feature	Raw Grayscale
Feature Bandwidth, σ	0.2
Training Rate	0.075
Spatial Bandwidth	$\frac{\sqrt{T_x \times T_y}}{10}$
Regularisation, λ	10^{-4}
Target Window Padding	2.5× target size
Scale Adaptation	True
Scales Tested	$[0.95, 1, 1.05]$
Scale Weights	$[0.95, 1, 0.95]$

STRUCK

Image Feature	Raw Grayscale
Kernel	Linear
Seed	20
Search Radius	30
SVM Regularisation, C	1.0
Support Vector Budget	100

Tracking on SPAD Datasets

Meanshift

Image Feature	Raw Grayscale
Quantisation Bins	16
Max Iterations	20
Adaptation Rate, γ	0.05
Scale Adaptation	False

KCF

Image Feature	Raw Grayscale
Feature Bandwidth, σ	0.2
Training Rate	0.075
Spatial Bandwidth	$\frac{\sqrt{T_x \times T_y}}{10}$
Regularisation, λ	10^{-4}
Target Window Padding	2.5× target size
Scale Adaptation	False

STRUCK

Image Feature	Raw Grayscale
Kernel	Linear
Seed	20
Search Radius	30
SVM Regularisation, C	1.0
Support Vector Budget	100

Appendix B MATLAB Test Bench Overview

Below is an overview of the main components of the MATLAB test bench tool that was developed in order to implement, run and evaluate the trackers.

B.1 Configuration

The configuration parameters are defined in JSON files inside the `tests` folder. These files define which trackers are used, overrides any default tracker parameters, and selects which datasets are to be used. In the case of SPAD data, this also specifies the feature to be used and any parameters of the feature.

B.2 Datasets

Each dataset is contained in a separate folder within the `datasets` folder. The `dataset.mat` file contains the metadata about the sequence, specifying the location of the image files, and the ground truth bounding box for each frame. For SPAD sequences, this file also contains the raw SPAD data.

B.3 Running Tests

Once a test configuration has been defined, it can be run using the `run_test` function. This reads the configuration file, initialises the trackers and load the datasets. The tracking process is then performed by loading each frame of the dataset and calling the `update` function of each tracker. The bounding box that each tracker outputs is then saved in a results file. A live video view with bounding boxes overlaid can optionally be displayed while the test is running.

B.4 Evaluating Results

Once a results file has been generated for a given dataset, the `evaluate_metrics` function is used to load the results as well as the ground truth for the sequence. For each frame, the results from all trackers are evaluated against the ground truth using the metrics specified. This is then used to generate the graphs of the metrics over time, and the precision and success curves.