Visual Tracking and Localisation For Drone Perception

Jonathan Munro

April 2017

Final year project thesis submitted in support of the degree of Master of Engineering in Computer Science and Electronics

Department of Electrical & Electronic Engineering
University of Bristol

Abstract

Drones are becoming popular for filming sporting events where a need to localise the drone and track athletes is of importance. However there are no benchmarking for tracking algorithms of sporting events shot by drones. This work has bench-marked a series of state of the art tracking algorithms that run in real-time against a novel data-set of collected sporting events videos filmed from drones. A C++ implementation of the Fast Discriminative Scale Based Tracker based on M Denelljan et al. algorithm [1], was the best performing tracking algorithm and was developed during this work. A potential solution to overcome the localisation problem has been proposed, ORB SLAM [2], and with help from Aristotle University of Thessaloniki, has been extended to save maps from previous flights to improve the accuracy of localisation during the filming of events.

DECLARATION AND DISCLAIMER

Contents

1 Introduction

The growth of UAV applications for consumer and commercial use is predicted to increase in the following years, the firm Lux Research estimates that the global market for commercial drones will reach $1.7 billion by 2025. [3] One application for these consumer UAVs is take aerial video footage for film, television and live sporting events. The article "Action Sports, Social Media, and New Technologies" [4] states that aerial drones are becoming common place for sporting events providing a unique perspective for an audience. It also states the growing use of drones among amateur sport enthusiasts as well among television networks such as ESPN using camera equipped drones during a live sporting events, the first of which was during the Winter X Games in Aspen Colorado.

Current top of the range technologies in this area for professional use include the DJI Phantom 3 Professional, YUNEEC Q500 4K Typhoon and 3D Robotics Iris+ [5]. These allow users to pre-plan flight routes using GPS and some allow tracking of a target however the functionality is still limited. If obstacles are in the way the UAV should know enough information about the world to adjust its path or if need be identifying places for safe emergency landing therefore reliable and accurate localisation system is needed. The UAV should also be able to track targets of interest in the environments (e.g. athletes during sporting events) to obtain good quality video footage.

A localisation technique, Visual Simultaneous Localisation and Mapping(SLAM) and 2D tracking algorithms have been studied to overcome challenges faced in the applications specified above. Both sets of algorithms, as well as producing accurate readings, should be efficient enough to run on a light mobile computing platform that the UAV can carry for at least 15 minutes of filming. Other technologies will also require computing power (e.g. obstacle avoidance algorithms) therefore algorithms proposed should run faster than real time.

1.1 Visual Tracking Problem

The tracking problem involves estimating the spacial and temporal changes of an object as it moves around the scene. This requires building a model of what is to be tracked and use this to give knowledge of where they object has moved in the subsequent frames. This differs from detection which determines whether an object is in the scene, sometimes locating it but only for the current frame.

It is important to understand the issues trackers face before evaluating current techniques. The main problems tracking algorithms need to overcome [6]are listed bellow stating their importance to aerial sports tracking application.

- Illumination Changes
- Background Clutter

- Deformations - Important as athletes being tracked are a non-rigid object
- Scale changes - Large scale variations will be apparent as both the target and camera are moving. The Camera will also need to change altitude to achieve different
- Abrupt changes in motion - Athletes will be accelerating and decelerating throughout so the tracker will need to be
- Camera Motion - UAVs will be moving requiring
- Occlusions - This includes partial occlusions, part of the object outside the frame and full occlusions.
- Similar Appearances - Athletes in teams will tend to look extremely.
- Different Shot types - UAVs provide a unique shot type compared to most benchmarking Visual Tracking datasets
- Time Complexity - Real time performance is needed due operating on a mobile platform online.

The design of a visual tracker focuses on the features extracted from images used to represent an object and how to detect this region of interest in later frames.

Existing tracking algorithms have not been bench-marked on UAV datasets of sport videos. Potential state of the art 2D trackers have been assessed for their suitability for UAV sporting applications. Assessment of visual tracking algorithms was the main focus of this work.

1.2 SLAM problem

Using GPS to localise a UAV is unreliable as a line of sight to a GPS satellite is not guaranteed, other solutions such as Inertial Measurements Units are also unreliable as errors in the readings can propagate as the time of flight increases. Simultaneous Localisation and Mapping (SLAM) algorithms can solve this problem. SLAM takes readings from the environment and localises the UAV with respect to these readings, thus creating a map of the environment and localising itself in it.

A SLAM solution has been chosen and adapted to load previous computed maps of an environment and basic tests of functionality. Due to the time constraints and lack of available resources to create ground-truth and in-depth study of SLAM solutions has not been carried out.

1.3 Report Structure

The 2nd section of this report gives a summary of feature descriptors used for Visual Tracking and SLAM; this is due to the overlap in some of the techniques. The 3rd section explains visual tracking techniques outlining the operation of common and state of the art algorithms.

Sections 4 and 5 explin how the unique Sports data-set was created and evaluated. Section 6 shows the design methodology for the implementation of the Fast Discriminative Scale Space Tracker. Section 7 and 8 provide analysis of the results of the benchmark and a conclusion.

A discussion of the work towards SLAM is provided in section 9.

2 Feature Extraction for SLAM and Visual Tracking

An important part of both Visual Tracking and SLAM is the type of features extracted from images. A summary of different feature descriptors are presented. This section provides an overview of different feature selection and extraction techniques.

Feature selection derives a set of values from an input (pixel values of an image) to reduce the size of a data set whilst retaining relevant information. Deriving further features from the feature space is called feature extraction. It is important to choose a set of features invariant to unwanted attributes in the scene (such as moving objects in SLAM and background clutter in visual tracking).

There are 2 different types of features/descriptors, local and global, which provide different types of information about an image. Global based descriptors provide information about a whole region of the image such as a contour map or a histogram whereas local based descriptors highlight key areas of an image. Global features tend to be more computationally efficient to compute than local features as well as producing a compact representation of the image [7], however are susceptible to global appearance changes such as light intensity of a scene or out of plane rotations [8]. Local descriptors have precise spacial information about key features but suffer from noise.

2.1 Local Features

Local features need repeatability (the same features can be extracted in multiple image), to be distinctive (Account for large variations), efficient (low time complexity to compute and compact enough to keep dimensionality of the data down) and give precise locality. A survey on local features [9] argues repeatability is the most important and a good feature is one that is invariant to large scale changes and deformations of an object.

Early local descriptors focused on edges and corners including the Harris corner detector [10] and the Canny edge detector [11]. An edge occurs when there in pixel intensity change in any direction in the image. Simple edge detectors find the magnitude of an intensity change and direction of the change in an image by performing convolution with 2 kernels, S_x and S_y, that approximate the derivatives in the x and y directions.

$$S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} S_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{1}$$

The magnitude,S, and direction,D, can then be calculated by the following equations.

$$S = \sqrt{S_y^2 + S_x^2} \qquad D = \arctan \frac{S_y}{S_x} \tag{2}$$

Canny edge detection improves on this with prepossessing and post-processing stages to remove weak edges and eliminate multiple edge values detections. Gaussian smoothing is used to eliminate noise prior to the Sobel edge detector. After Sobel edge detection, non-maximum suppression eliminates spurious edges as well as evaluating a hysteresis threshold to remove low intensity edges. [12]

With a highly textured video with viewpoint changes, edges are not a distinctive feature and are difficult to match successfully in different frames. In these cases Corners are a more appropriate feature.

Corner detection calculates the sum of squared differences between an image and a small shifts by (uv) in all directions. If a corner is present there will be a large intensity change in all directions.

$$H = \begin{bmatrix} u & v \end{bmatrix} \left(\sum_{x,y} w(u \quad v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix} \tag{3}$$

The Harris matrix, as shown above, approximates this value. The partial derivatives I_y and I_x need to be calculated and a Gaussian weighted function $w(u, v)$ is to be applied to make the result isotropic (change in pixel intensity in all directions). As corner detection is isotropic they are more distinctive than edges and can be matched more reliably in subsequent frames.

Harris corner detection is not invariant to scale variation. David G. Lowe paper on Scale Invariant Feature Transformation [13] provides a scale and rotation invariant point detector. The primary stages of this algorithm are: scale-space extrema detection that provides key points invariant of scale and orientation assignment to the descriptor making key points invariant to rotation.

Scale-space extrema uses Difference of Gaussian(DoG) as a blob detector of corners and edges at different scales. A set of standard deviation values are grouped into octaves. The equation bellow is used to calculate DoG where I is the image, G is a 2D Gaussian function and σ_1 and σ_2 are neighbouring standard deviation values within an octave.

$$DoG = I * G(x, y, \sigma_1) - I * G(x, y, \sigma_2) \tag{4}$$

By changing σ of the Gaussian filters, corners at different scales can be detected. Key points are only chosen if they are a maxima or minima of both scale and position in a given octave. A hessian matrix can be convolved with an image to find the local curvature. Key points are removed if the local curvature is less than a given threshold to eliminate edges and low intensity corners. Orientation and magnitude can be calculated for each chosen key point by equation 2 where S_x and S_y are the difference of neighbouring pixel intensities in the x and y axis.

Although SIFT features are popular that they are inefficient to compute. Alternatives such as SURF [14] and in more recent years Orientated Rotated Briefs (ORB) [15] have found key points more efficiently.

2.2 Global and Region based Features

Global descriptors provide information about a whole area of an image (e.g. an object to be tracked) differing from local descriptors that can provide information about key points. These features are useful for visual tracking to create models about objects but are not generally used for SLAM algorithms as accurate distances from features are needed. A basic example of a global descriptor would be the set of raw pixels but this provides a large amount of information that are not invariant for a given object.

Histograms are a popular global descriptors that provide a count of how many input values fall within given intervals (bins). A simple example would be an intensity histogram that provides the number of pixels that intensity fall within a given bin. This method is prone to global intensity changes therefore is often normalised by dividing each count by the total intensity of an image. Colour Histograms are a common descriptor for tracking algorithms providing intensity histograms for the 3 colour channels of an image (usually in RGB format).

Histogram of Orientated Gradients (HOG) [16] combines local edge information (change in intensity in a given direction) by providing a summation of the magnitude of all edges that falls within a series of orientation bins. Rectangular HOG (R-HOG) evaluates a histogram for a series of square sections of an image called cells. The cells are then grouped into blocks and histograms are normalised. The figure bellow shows a visualisation of the hog descriptor, where the length of the white lines represent the magnitude of the histogram in a given orientation.

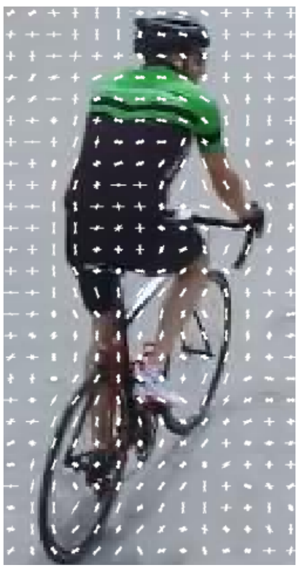


Figure 1: Matlab Visualisation of a HOG descriptor (cyclist4 given in Appendix C)

Haar-like features have been used by various tracking algorithms and first proposed by Viola and Jones [17]. An example of Haar features are provided bellow. The features are iterated across an image patch taking the difference between the sum of the pixel values in the black and white regions to obtain descriptor values.

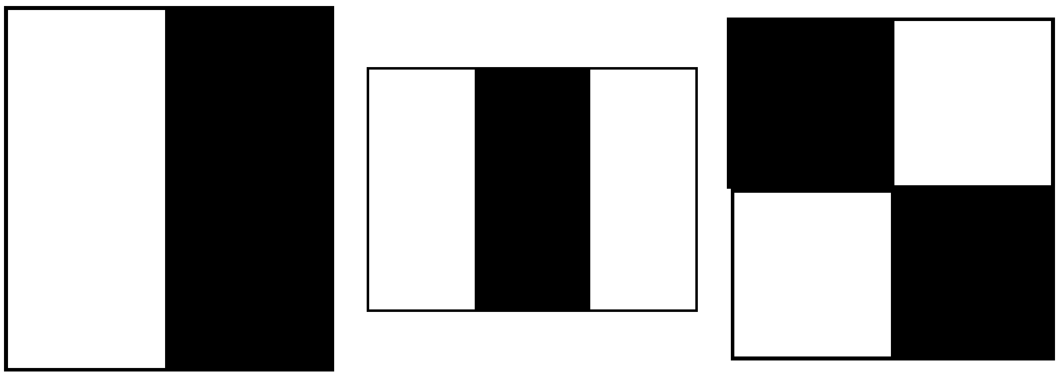


Figure 2: Example of Haar Features

3 Current Visual Tracking Methods

Visual tracking algorithms require: a set of features ideally invariant to the objects appearance, an object representation built from those features and a classifier to evaluate the position of the represented object in subsequent frames. A summary of different feature descriptors was provided in the previous section explaining the variety of sparse local features and dense global features that can be used. This section focuses on object representation and classification techniques.

3.1 Object Representation

Object representations can be classified as discriminative or generative. Discriminative representations are created by maximising the distance between 2 classes (in this case the image region containing the object to be tracked and areas that do not). Discriminative models learn a conditional probabilistic model whereas generative models build a model of the object from the features available creating a joint probabilistic model. Discriminative representations tend to be more sensitive to noise however generative ones are more sensitive to clutter [18].

Some models have tried to reduce the impact of the background information for example the Backward-Weighted Histogram (BWH) [19]. BWH defines the background as the surrounding area that is 3 times larger than the target. A colour histogram object representation is weighted such that similar pixel values to the reference background impact is reduced. Given the background features $\sum_{u=1}^{m} \hat{o}_u$ and the smallest non-zero feature in the background $\hat{o}^*$ the weighting vector is defined by

$$v_u = min(\frac{\hat{o}^*}{\hat{o}_u}, 1) \tag{5}$$

Object representations are usually evaluated on the first frame passed to the tracker where the object region is known but can be updated during subsequent frames. Updating the object representation can provide more accurate results by removing feature points relating to clutter and adapting to deformations of the object however this can lead to drift.

3.2 Classification Methods

3.2.1 Discriminative Models

Early Discriminant trackers commonly used Boosting methods, such as Ada-Boost used in [20] [21], for classification. Boosting algorithms combined a series of weak classifiers (classification success only slightly better greater random) to create a strong classifier that group pixels to target and non-target regions. Amongst more recent state of the art methods use Support Vector Machines (SVM) [22] or Discriminant Correlation Filters(DCF) [23] [24].

Correlation Filters
DCF creates a correlation function h that minimises the squared difference between a set of N input features $\sum_{l=1}^{N} f^l$ of the a region containing the target (ROI) and the desired correlation output g. The desired output is usually a Gaussian. λ is a regularisation parameter.

$$argmin_h \left(|| \sum_{l=1}^{N} h^l * f^l - g^l ||^2 + \lambda || \sum_{l=1}^{N} h^l ||^2 \right) \tag{6}$$

The advantage of using DCF in a discriminative tracker is that it can be solved by least squares efficiently in the frequency domain. As correlation in the spacial domain is point-wise multiplication in the frequency domain the time complexity is reduced from $O(N^2)$ to $O(NlogN)$.

$$H^l = \frac{\overline{G} F^l}{\sum_{k=1}^{N} \overline{F^k} F^k + \lambda} \tag{7}$$

The discriminant model, H, is split into it's numerator, A, and denominator, B, and updated for subsequent frames. A learning rate parameter η controls the impact of later frames on the model.

$$A_t^l = (1 - \eta) A_{t-1}^l + \eta G^l F t^l$$

$$B_t = (1 - \eta) B_{t-1} + \eta \sum_{k=1}^{N} \overline{F_t^k} F_t^k$$

Given an image patch Z extracted from the current frame with the coordinates of the previous frame's ROI, the centre of the current frame's ROI is predicted as the coordinate that maximises y.

$$y = F^-1 \frac{\sum_{l=1}^{N} \overline{A^l} Z^l}{B + \lambda} \tag{8}$$

3.2.2 Generative Models

Most Generative tracking algorithms are classified using the Bayes rule.

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} = \frac{p(x,y)}{p(x)} \tag{9}$$

$p(x, y)$ is the joint probability that of obtaining a set of features and the image patch belongs to a given class (object or non-object). In most applications it can be assumed that the evidence (probability of obtaining a set of features) $p(x)$ is constant. The posterior generated specifies the probability of correctly predicting the location of the target given a set of features. Image patches of frames that maximise the posterior probability are interpreted as the new target position.

A popular implementation of the Naive Bayes classifiers with visual trackers are particle filters. A set of samples(particles) are used to model a

potential state of the target. The samples and weights according to the target likelihood that it belongs to that sample. The particles are re-sampled according to their weights and propagated using a motion model of the target. A large number of particles are required for accurate target tracking which can be much slower than other forms of generative techniques and can still suffer from the sample impoverishment problem [25].
Other options for generative tracking algorithms include part based algorithms that match sparse local features such as SIFT as well as global appearance based ones such as mean shift tracking [26].

Mean Shift Tracking

Mean shift tracking (MSF) minimises the distance between 2 probability density functions (pdfs) : an object model and a candidate. Both pdfs are derived from colour histograms features.

The object model for MSF [27]is a vector stating the probabilities a given colour bin,$u = 1...m$, belongs to the object. Given the objects image patch features,$\sum_{i=0}^{n} x_i$, the object model is obtained by the bellow equation. K is a kernel function that is typically a Gaussian and b maps a colour to its respective histogram bin.

$$q_u = C \sum_{i=1}^{n} k(||x_i^2||)\delta(b(x_i) - u) \tag{10}$$

A Candidate pdf [27] for an image patch's features centred at y can be obtained by evaluating the bellow equation. This creates a probability vector for each colour bin belonging to the target.

$$p_u(y) = C_h \sum_{i=1}^{n_h} k(||\frac{y - y_i}{h}||^2)\delta(b(y_i) - u) \tag{11}$$

look at [28]

3.3 State of the Art

Discriminant Scale Space Tracking [29] has been proven one of the State of the Art tracking algorithms that are based on correlations filters. A correlation filter is used to predict the target's centre coordinate and the scale is evaluated by another correlation filter. DSST extends earlier correlation filter techniques such as MOSSE [23] by:

- Adding a grey-scale interpretation of frames as features as well as the existing HOG features used.
- Adding an additional correlation filter to predict the scale of the target.

In recent years algorithms such as Spatially Regularized Discriminative Correlation Filters(SRDCF) [30] have improved on DSST. SRDCF has introduced spacial regularisation by changing the regularisation parameter λ in the correlation filter. This reduces the boundary effects induced by circular convolution. However despite added performance, these algorithms do not run in real time and therefore not suitable for UAV applications.

STRUCK [22] is another state of the art discriminant tracking algorithm. This technique used Support Vector Machines to maximise the distance between target and non-target regions.

Adapative Scale Meanshift Tracking (ASMS) is a generative tracker that use a mean-shift classifier using colour histogram features. ASMS improves upon the mean-shift classifier by:

- Reduces the impact of background features by incorporating an improved BWH algorithm.
- Scale adaptation

Amongst the best performing tracking algorithms deep learning has been used to create feature descriptors however the run-time of these algorithms are slow (generally $fps < 5$)

4 UAV Bencharmk Data-Set for Sports Creation

4.1 Existing Tracking Data Sets

To determine the performance of tracking algorithms a series of benchmark videos are required with annotations specifying the bounding boxes encapsulating the object to be tracked. Existing data-sets provide a large number of short videos primarily focusing on typical problems tracking algorithms face such as partial occlusion and background clutter. However there are a limited number of benchmarks providing unique shot types that are produced by drone footage. The UAV123 dataset by Mueller et al. [31] provided a unique collection of videos exclusively filmed from UAVs. However these videos were not dedicated to a particular application. This work creates a data-set of videos from UAVs perspective of a variety of sporting applications.

4.2 Data set collection

A selection of videos containing a variety of sporting environments were collected. These included cyclists, footballers, boats, wake-boarders and rowers. These videos were obtained from online authors that uploaded their videos to YouTube [32] [33] [33], from the UAV123 dataset [31] and others provided by the Arrisotle University of Thessaloniki (AUTH).

4.3 Ground Truth Creation

An annotator tool provided by AUTH was used to annotate frames by hand. To speed up the process the STRUCK tracker [22] was integrated into the tool and set to run until errors in tracking occurred. To create accurate ground truth data the tracking algorithm had to be stopped often (on average every 15 frames) and ground truth corrected. Due to the large numbers of frames help annotating was provided by students at AUTH however the majority of annotation (excluding already annotated videos provided by UAV123) was still done myself.

5 Visual Tracking Evaluation using Sports benchmark

5.1 Evaluation Metrics

The metrics used to evaluate the tracking algorithm are Success and Precession used in Wu et al. paper [34] For a targets ground truth box, ROI, and the predicted target box, ROI', a measure of overlap is given by the Success

$$Success = \frac{|ROI \cap ROI'|}{|ROI \cup ROI'|} \tag{12}$$

Precision is defined as the euclidean distance between ROI and ROI' center points, (x,y) and (x',y')

$$Precision = \sqrt{(x - x')^2 + (y - y')^2} \tag{13}$$

The Precision rate is the ratio of frames with Precision less than an location error threshold.The Success rate is the ratio of frames with Success greater than an overlap threshold.

5.2 Method

Tracking algorithms were initialised to the first ground-truth ROI available. The algorithm was left to run attempting to track the target. For each frame the Success and Precision was calculated by comparing the predicted result with the annotated ground-truth. The tracking algorithm was never re-initialised after the first frame.

5.3 Selected Algorithms

The following Algorithms have found to be amongst State of the Art methods that run in real time:

- Adaptive Scale Mean Shift (ASMS) [35]
- Discriminative Scale Space Tracking (DSST)
- Structured Output Tracking with Kernels (STRUCK)
- Fast Discriminative Scale Space Tracking (FDSST)

ASMS, DSST, STRUCK all had open source C/C++ implementations however only open source implementation of FDSST available prior to this project was written in Matlab. Matlab is unsuitable for most UAV applications as the tracking algorithm may need to be incorporated into the ROS (Robotics Operating System) as well as other systems that do not have to rely on Matlab being installed. Other disadvantages is the high cost of Matlab for industrial applications as well as the reduced runtime due to being an interpreter. A C++ implementation of FDSST was developed and was the implementation that was evaluated.

6 C++ Implementation of the Fast Discriminative Scale Based Tracker

A promising tracking algorithm (FDSST) extends DSST by improving both the performance and run-time of the algorithm. The decreased run-time is due to the reduced number of Fast Fourier Transform (FFT) operations used to create the object model H. The original author also increased the number of scales the algorithm compares that lead to an increased performance.

6.1 Overview of FDSST Algorithmic Extensions to DSST

Principle component analysis (PCA) of the objects HOG feature space, f, is used reduce the number of FFTs when creating a object model, H. A new target template, u, is updated for subsequent frames instead of the numerator A. The target template is initially set to objects features on the first frame as the objects position is known.

$$u_0 = f_0$$

$$u_t = (1-\eta)u_{t-1} + \eta f_t \tag{14}$$

A Projection matrix p_t was used to map the feature spaces to a reduced set of features defined by the principle components of the target template u.

- An autocorrelation matrix, C is produced

$$C_t = \sum_n u_t(n)u_t(n)^T \tag{15}$$

- A series of ranked Eigenvectors of C can be found using Single Value Decomposition.
- p_t is created by choosing the first d eigenvectors of C

The new numerator and denominator of the object model, A and B are defined below:

$$A_t^l = \overline{G}\mathcal{F}(p_t u')^l \tag{16}$$

$$B_t = (1-\eta)B_{t-1} + \eta \sum_{l=1}^{N} \overline{\mathcal{F}(p_t f')^l}\mathcal{F}(p_t f')^l \tag{17}$$

Given a new image patch z the targets central location is at the coordinate that maximises y

$$y = \mathcal{F}^{-}1 \frac{\sum_{l=1}^{N} \overline{A^l}\mathcal{F}(p_t z)^l}{B+\lambda} \tag{18}$$

6.2 Profiling DSST

For DSST the larger the target area the slower the runtime. The graph bellow relates the targets size against the percentage run-time FFT and spectrum multiplication contributes to the total. This shows the potential benefits of FDSST at speeding up the run-time of tracking with large targets compared to DSST where the run-time is less likely to be real-time.

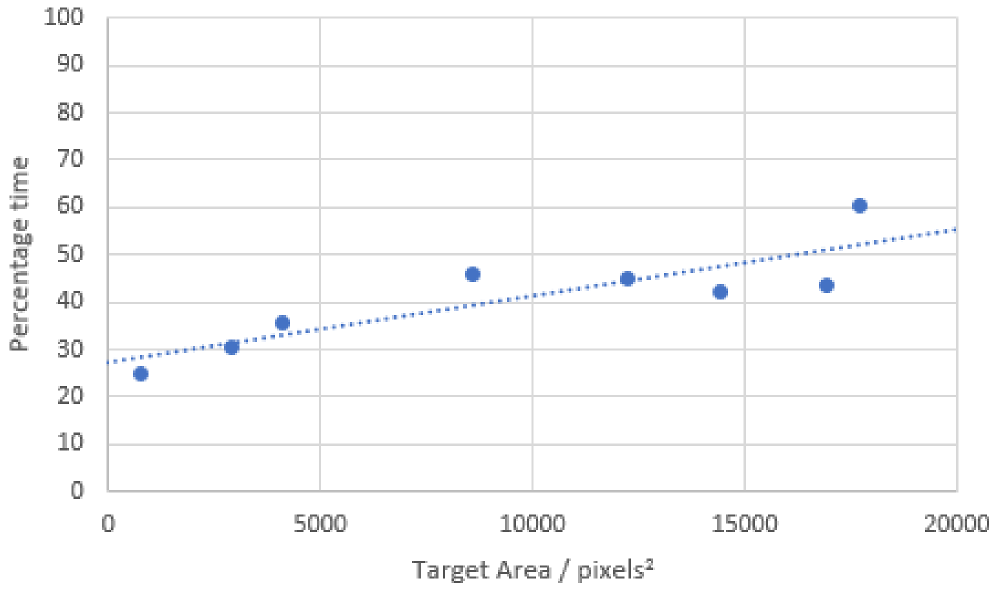


Figure 3: Percentage time FFT and spectrum multiplication contributes to the total time of an DSST tracker

6.3 Design Considerations

6.3.1 Frequency Domain Represenation

To reduce the runtime of processing complex numbers in the frequency domain all complex data types used the Intel's complex-conjugate-symmetrical format [?] that exploit conjugate symmetry. This allows half of the numbers int the frequency domain to be processed.

6.3.2 Greyscale Image Feature with Increased Cell Size

For FDSST cells of 2x2 pixels are used for each block in calculating the HOG features. To keep all features the same length, grey-scaled image feature must be down-sampled by a factor of 2 in both the x and y directions. To do this an average pixel value for each cell was to be calculated. This was efficiently done using a method called Integral Image.

Given an image i a matrix I was computed in a single pass.

$$I(x,y) = \sum_{x'<x,y'<y} i(x',y') \tag{19}$$

The new down-sampled greyscale image feature, G, is defined bellow.

$$G(x,y) = \frac{I(x,y) + I(x-2,y-2) - I(x,y-2) - I(x-2,y)}{4} \tag{20}$$

6.3.3 Toolkits

OpenCV was chosen as the framework for the FDSST implementation. To produce fastest results the library was compiled using Intel Performance Primitives that provide optimised subroutines with AVX vector extensions enabled.

6.3.4 Testing

The implementation was designed such that functions carried out small tasks and were compared with equivalent tasks in the original Matlab version acting as a model for unit testing. Each of these functions was then integrated and tested against the Matlab implementation.

6.4 Choosing Parameters

Parameters that would maintain a good estimate the objects location while allowing a fast run-time were selected. Decreasing the learning rate of the model allowed FDSST to support long-term tracking.

6.4.1 Decreased learning Learning Rate for Long Term Tracking

The learning rate, η controls the impact of later frames on the object model. The original DSST paper uses $\eta = 0.025$, however using this value causes significant drift making it unsuitable for long term tracking. A paper by D S Bolme et al. [23] found that $\eta = 0.0125$ was sufficient to adapt to new object appearances while maintaining a robust tracker.

Testing Long Term tracking

To test the reduced η value effectiveness at long-term tracking, a series of non-annotated rowing videos were collected. The tracking algorithm with $\eta = 0.025$ fails within 3 minutes of tracking. However for all 3 videos FDSST with $\eta = 0.0125$ is successful throughout (average length of video was 7 minutes).

6.4.2 Increased Search Range for Robustness

To be able to detect objects that move outside the predicted object region (ROI) of the previous frame, ROI must be padded to include surrounding regions. The set of features of the object, f, can then be calculated from the image patch defined by the padded object boundary (ROI_f)

$$ROI_f = ROI(1 + padding) \tag{21}$$

It was suggested in M Denelljan et al. [1] that increasing the search range could improve the spacial correlation filters robustness. However with an increased search range a larger portion of the background is added to the model as well as an increased run-time. Experiments were performed on the Sports dataset with different search ranges for FDSST to find value that is robust and computationally efficient.

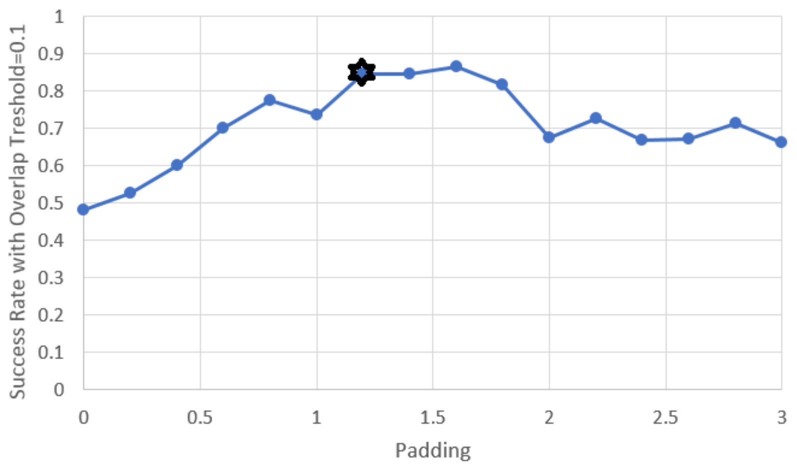


Figure 4: How the search range affects the tracking success

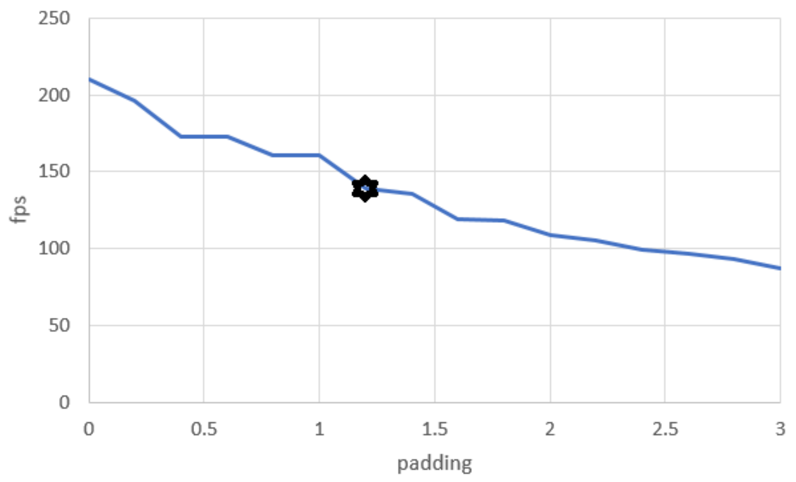


Figure 5: How the search range affects the frame rate

From the results it is clear that up to a padding value of 1.2 the number of tracker failures reduces with an increase in search range. Beyond 1.6 too much background is added to the model causing the number of failures to increase. A padding value of 1.2 was chosen for FDSST to allow a fast run-time as well as keeping a robust tracking algorithm.

6.4.3 Number of Principle Components

A number of principle components for the object model must be chosen that decreases the run-time but does not hinder tracking results. The graph bellow shows results of different number of principle components used for FDSST operation on the Sports benchmark. It can be seen that approximately 14 or more principle components, performance of the tracking algorithm is maintained. An experiment of effect the number of principle components had on the frame rates was also performed. The mean frame-rate of FDSST on the 36 videos of the Sports benchmark was calculated. It can be seen that reducing the number components increases the frame rate significantly. A value of 16 was a good trade off of keeping a robust tracking algorithm as well as a fast run-time.

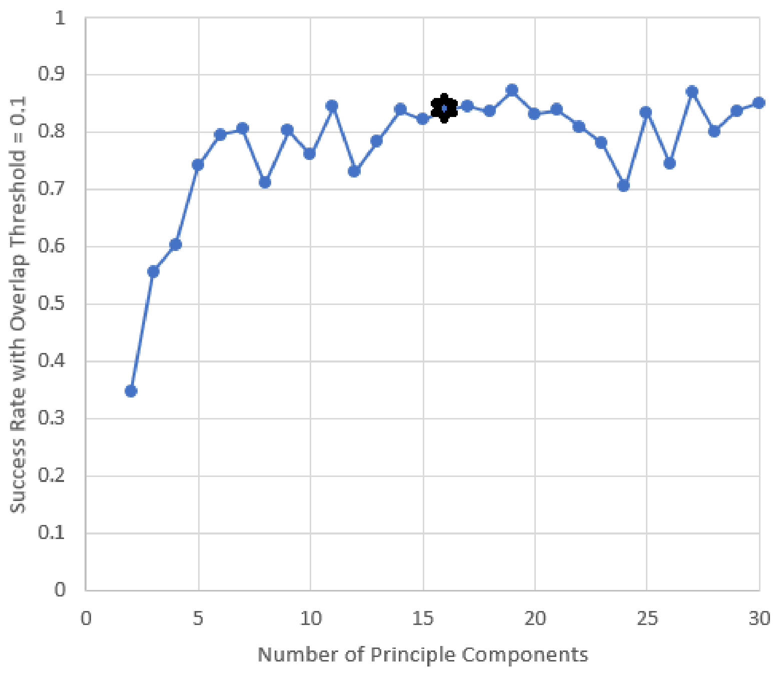


Figure 6: How the number of principle components affects the tracking success

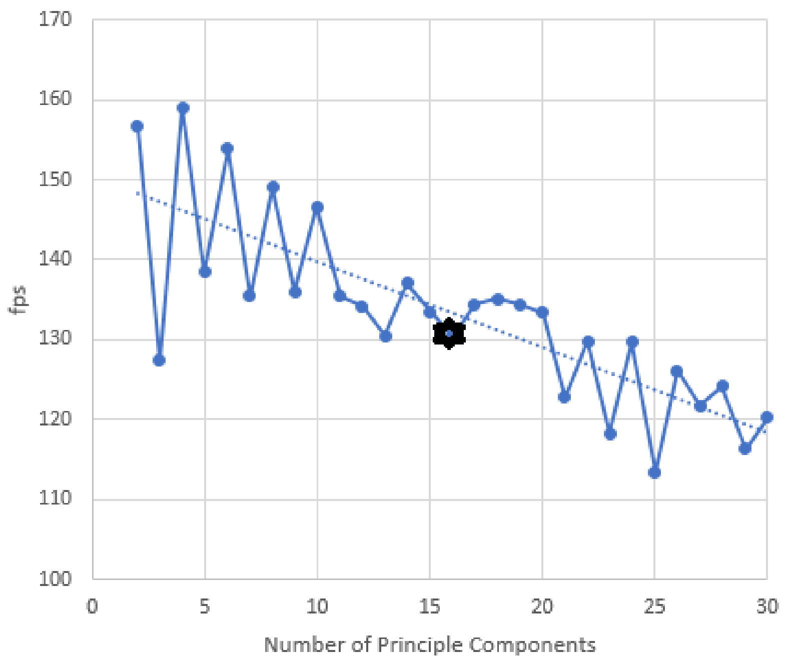


Figure 7: How the number of principle components affects the average frame rate

6.5 Comparisons with Matlab Implementation

With the same parameters the Matlab and C++ implementation of the algorithm gave the same success and precision performance however the run-time was significantly faster for the C++ implementations giving almost a 2x speed up for larger targets and a 10x speed up for small targets.

Video	Matlab fps	C++ fps
cyclist4	17.0	30.3
rowing1	13.4	24.8
football1	27.1	213.7

Table 1: Framerates of C++ and Matlab implementations of FDSST

The table bellow displays the difference in parameters values used in the FDSST's original paper [1] and the C++ implementation

Parameter	Matlab	C++
Learning Rate	0.025	0.012
Padding	1.6	1.2
Principle Components	18	16

Table 2: Framerates of C++ and Matlab implementations of FDSST

7 Analysis of Sports Benchmark Results

7.1 Overall Results of SPORTS Benchmark

FDSST outperformed all trackers by number of failures, success and precision. Despite STRUCK providing the 2nd least number of failures with higher success rates the overlap was smaller than all other tracking algorithms (the success values close to 1 were ignored for STRUCK as the ground-truth was created with this algorithm).

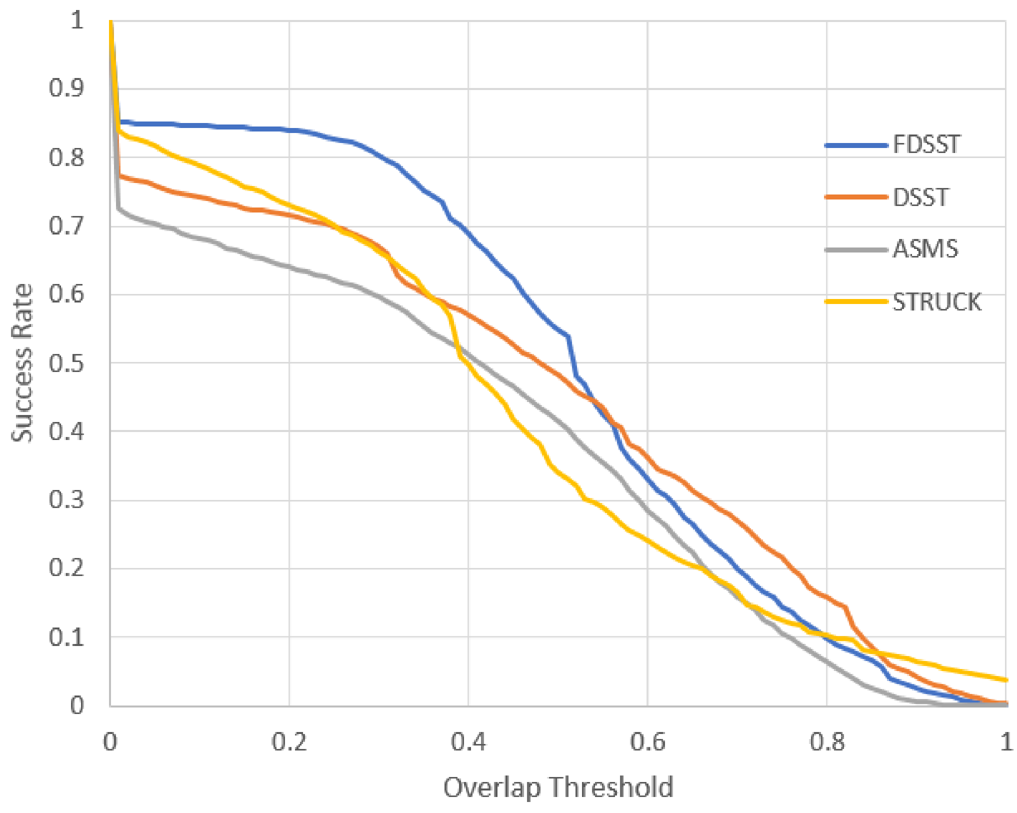


Figure 8: Success Plots

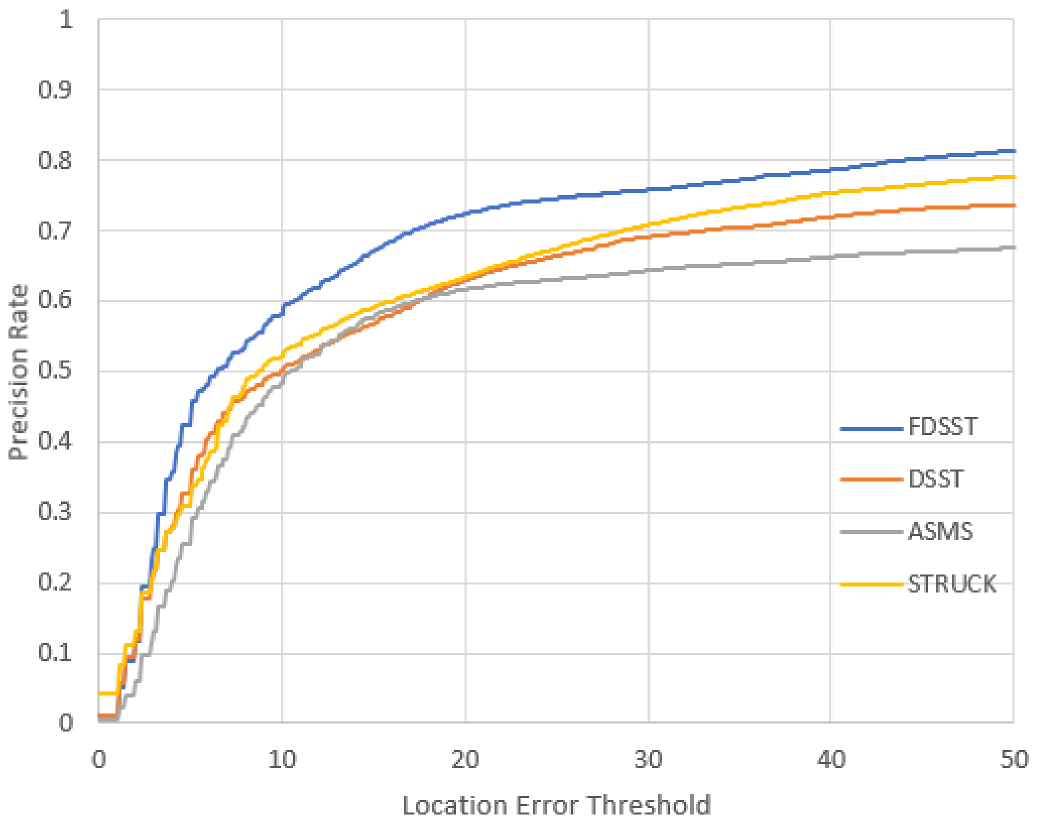


Figure 9: Precision Plots

The algorithms have been ranked in order of their success at an overlap threshold of 0.1. The lower the percentage the higher the rank. The average Success and Precision for each tracker is also provided.

Rank	Algorithm	Success Rate with Overlap Threshold=0.1	Mean Success	Mean Precision
1	FDSST	0.846	0.472	84.0
2	STRUCK	0.793	0.397	93.0
3	DSST	0.746	0.431	123.2
4	ASMS	0.685	0.365	148.5

Table 3: Ranking of Visual Tracking Algorithms

7.2 Results with an isotropic background

Although ASMS did not perform well over all, under an isotropic background (in this case a small object in-front of a large blue sea background) it out performed all other algorithms.

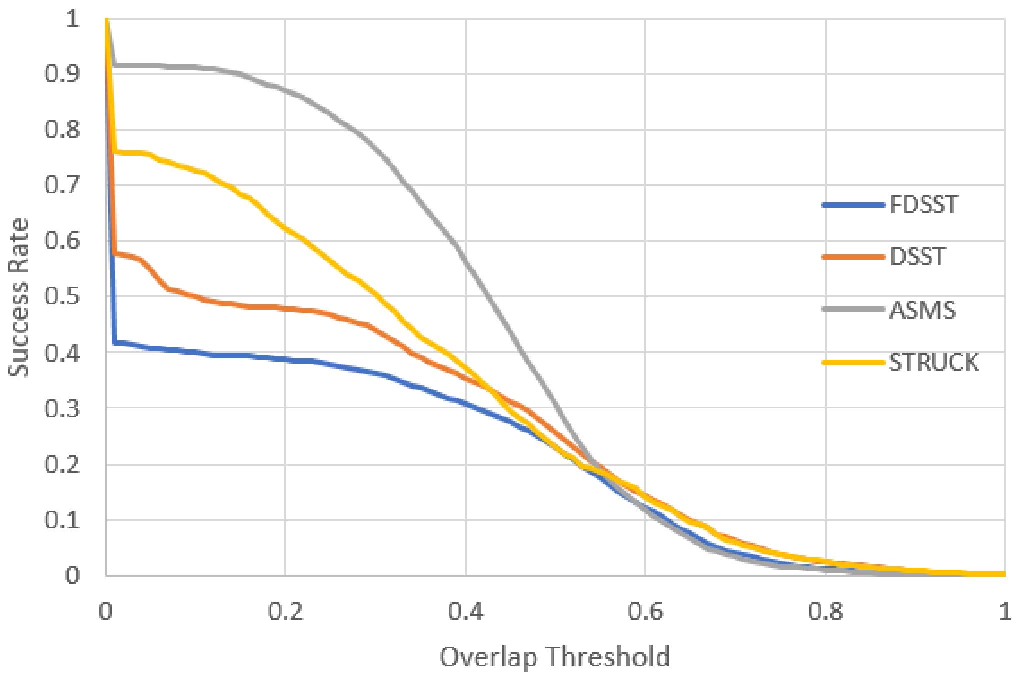


Figure 10: Isotropic Results

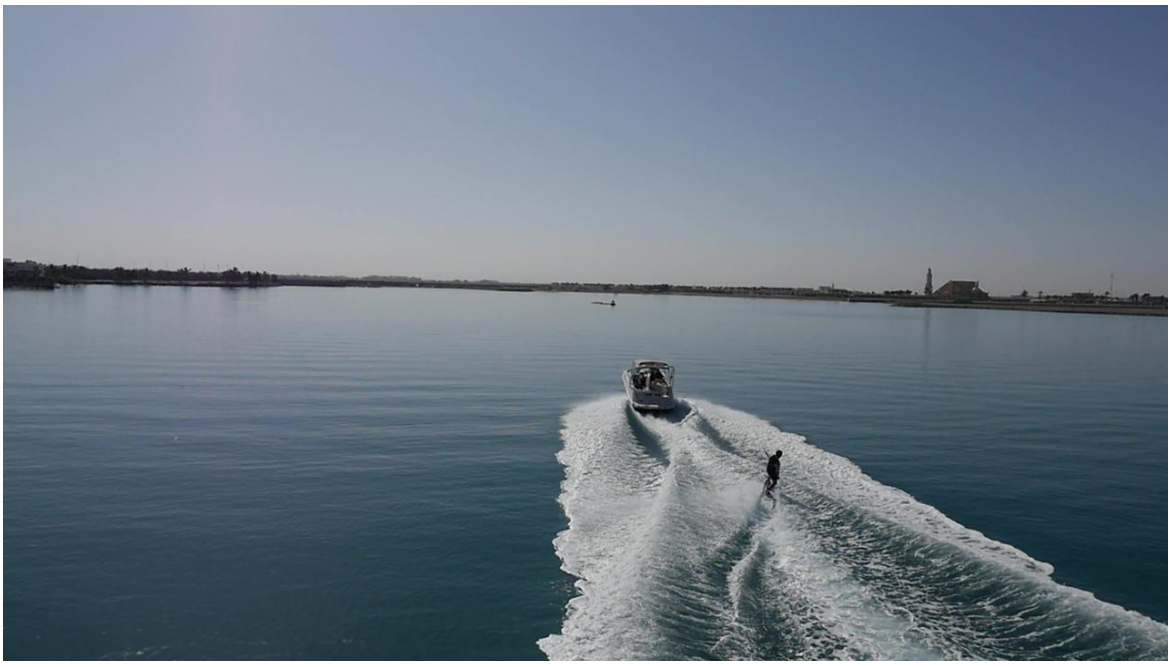


Figure 11: Example of An Isotropic Background Test Case (wakeboard2 given in Appendix C)

7.3 Qualitative Analysis

Illumination Changes
With large illumination at initialisation FDSST,DSST and ASMS fail almost immediately. STUCK can handle this with little loss of accuracy.

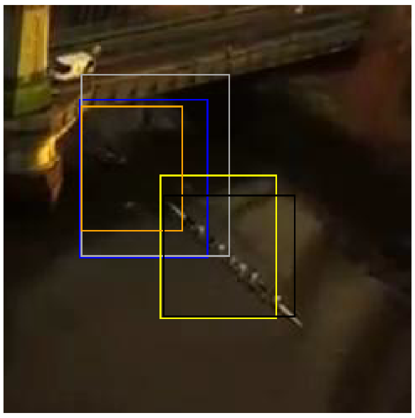


Figure 12: Illumination Change at Initialisation (rowing1_2 given in Appendix C)

Scale Changes
Only FDSST and DSST perform well against scale changes. ASMS although adapting to scale changes does not have the same success at DSST or FDSST. STRUCK has no scale adaption, and in some cases can cause it to lose track of a target.

Similar Appearances
DSST, ASMS and DSST struggled with similar appearances, the worst of which was ASMS. FDSST handled similar appearances well with little drift from the target

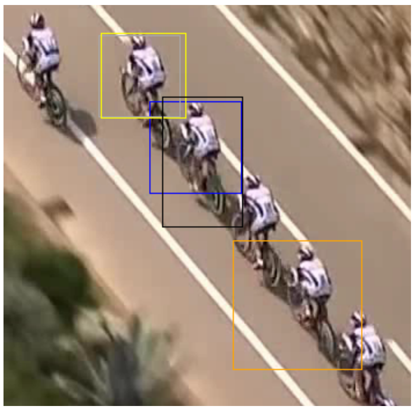
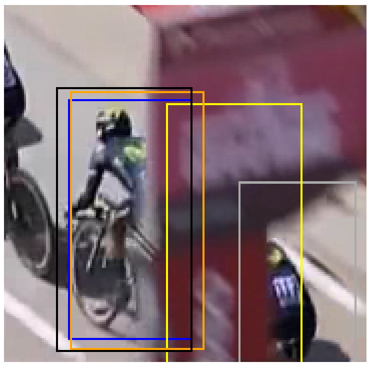


Figure 13: Similar Differences (giro1 given in Appendix C)

Occlusions
FDSST and DSST both handle occlusions well whereas STRUCK and ASMS struggle.


(a) After full occlusion (giro1 given in Appendix C)

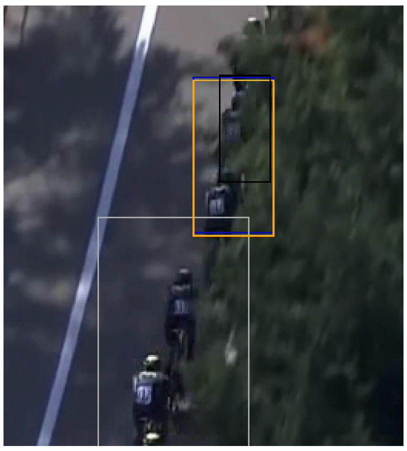

(b) Partial Occlusion (giro2 given in Appendix C)

Figure 14: Occlusions

Deformations
All Tracking algorithms could handle deformations of the target such as out of plane rotations. However despite good tracking accuracy the tracking algorithms did not adjust the ROI shape to match the deformed object
Target and Camera Motion
With an isotropic background FDSST and DSST struggled to follow the target. STRUCK could handle this better but not as well as ASMS

7.4 Tracking Speed

ASMS was the only garanteed real-time tracking algorithm that was benchmarked. However FDSST and DSST did provide a real-time solution as long as the tracker was initialised with a sufficiently small ROI. This was approximately 3000 $pixels^2$ of DSST and 20000 $pixels^2$ FDSST. In the majority of cases the ROI was not larger than 20000 (141x141) therefore FDSST can still be used as a real-time solution.
All results used the same computer to test the performance. This was an Intel Core i5-3230M CPU that has 2 cores.

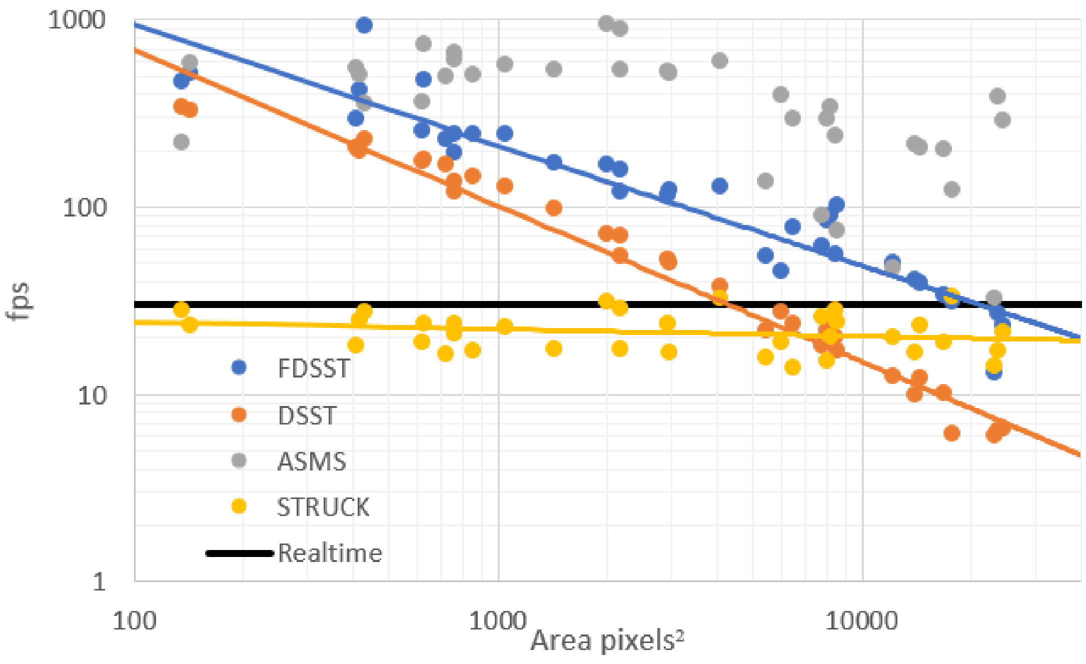


Figure 15: Framerates of Sport benchmark

8 Conclusion Sports Benchmark Results

The C++ implementation of a fast Discriminant Scale Space tracking algorithm (FDSST) has been proven to have better performance on the created sports data-set than other state of the art real-time tracking algorithms available (STRUCK, ASMS, DSST) making it best suited for use in tracking athletes from UAVs. FDSST runs in real time if the area of the target to be tracked is less than 20000 pixels which is adequate for most applications whereas current implementations of STRUCK and DSST have a slower framerates for most scenarios. Despite ASMS running faster than the other algorithms the performance on the Sports benchmark is significantly worse than the other tracking algorithms on average. Under isotropic backgrounds ASMS is less prone to tracking loss therefore future work could focus on combining ASMS and FDSST results.

9 Simultaneous Localisation and Mapping

9.1 SLAM methods

Visual SLAM algorithms solve for a map and a camera location given a series of frames. Among the earliest methods to solve this was the Kalman Filter. [36]. This predicts a state (location of camera) given a series of noisy observations (frames). The filter first predicts a new state based on a model and a map of the environment. A series of observations are collected and compared with the expected observation values of the new state. If the expected observations matches the actual observations (within a given uncertainty calculated by the model) the model can be refined using the actual observations. If there is no match it can be assumed that the map has changed and the observations are used to update the map. Note the Kalman filter can update a map as well as find the position of an object.

The issue with a Kalman filter is that the predictions are based on linear estimates more effective methods include Particle filters and the Extended Kalman filter that give non-linear estimates.

As more observations are taken the uncertainty of the location of observed features in the map will grow. A Loop closing algorithm detects if current readings have already been observed, this allows the map to correct all reading from the previous observation to the current observation reducing the uncertainty. This gives SLAM an advantage over other methods.

9.2 State of the Art

State of the Art methods tend to focus on improving the features used for observations. These features are local features as precise locality of the observations are needed. A novel SLAM approach ORB SLAM [2] uses ORB features to provide a real time solution on a CPU. ORB SLAM has monocular and stereo implementations [37] therefore has the possibility of extending to multiple UAV camera views with further research. Other solutions that have 3rd party monocular and stereo implementations include LSD-SLAM. [38] [39]

9.3 Extend ORB SLAM to use existing maps

ORB SLAM has a set of key-frames and a map are stored in self contained classes. To save maps from previous algorithm executions the state of the map and key-frames objects will need to be generated. Serialisation, provided by the Boost library, can be used to deconstruct the state of the classes into a sequence of bytes. At the beginning of execution the serialised objects are loaded into new key-frame and map objects and at the end of execution their state is saved into a file.

9.4 Conclusion

An existing SLAM solution has been extended to load maps of an environment prior to filming sporting events. However it is not known if this solution will be suitable to guarantee reliable location accuracy. Further work in creating ground truth for SLAM is needed to be able to test this. Without the necessary resources of a UAV and pilot it was not possible to create ground-truth for use in this project.

A Software Used

A.1 Source code

Source		Author	Descirption
ORB SLAM	C++	opencv.org/	modified by myself and AUTH
OpenCV3.2	C++	opencv.org/	Unmodified
DSST	C++	github.com/ klahaag/cf_tracking	Modified by a PHD student
ASMS	C++	github.com/ vojirt/asms	Unmodified
STRUCK	C++	github.com/ gnebehay/STRUCK	Unmodified
FDSST	Matlab	www.cvl.isy.liu.se/ en/research/objrec/ visualtracking/ scalvistrack/	Unmodified
FDSST	C++	own work	Modified DSST (see above) modified: dsst_tacker.hpp (fdsst_tracker.hpp) dsst_tacker_run.hpp (fdsst_tracker_run.hpp) dsst_debug.hpp (fdsst_debug.hpp) scale_estimator.hpp (fscale_estimator.hpp) feature_channels.hpp
LSK TrackerDLL	C++	Given by Supervisor	unmodified
STRUCK TrackerDLL	C++	Own work	used format of LSKTrackerDLL to integrate STUCK into AUTH annotator tool
Test Script	C++	own work	Benchmark Source Code

A.2 Software Tools

Visual Studio 2014 and 2015
Intel Parallel Studio XE 2017
AUTH annotator Tool
Matlab

B CD Structure

A CD has been provided with the thesis that includes the results and code of thesis project. The figure bellow shows the hierarchy of folders in the CD. The code is grouped into 3 folder for different work that I have modified or created and described in the section above.

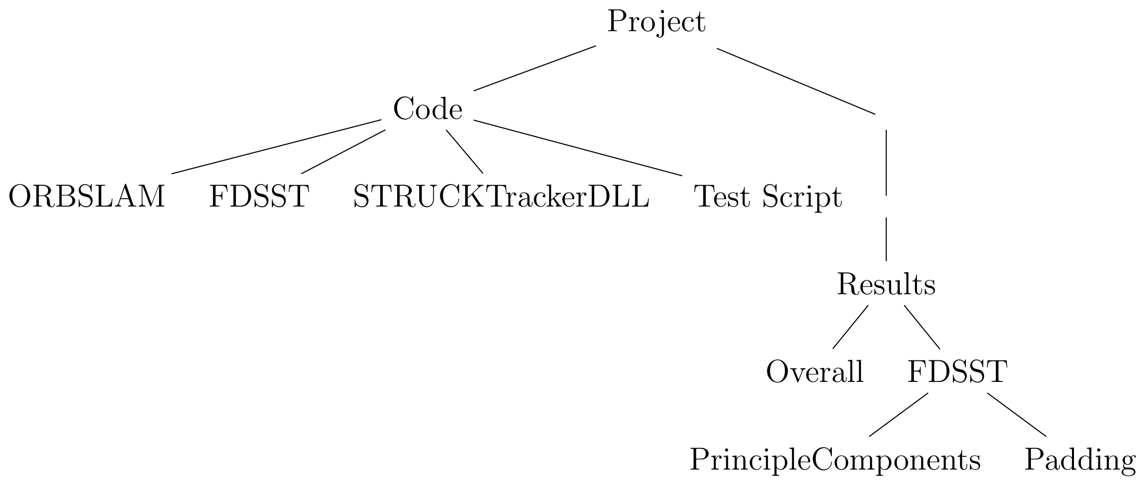


Figure 16: Directory Structure of Attached CD

C Sports Data-set

Video	Source	Annotation Work	Frame Dimentions	Isotpropic
bike1	UAV123	UAV123	1280x720	
bike2	UAV123	UAV123	1280x720	
boat1	UAV123	UAV123	1280x720	
boat2	UAV123	UAV123	1280x720	
boat3	UAV123	UAV123	1280x720	
boat4	UAV123	UAV123	1280x720	
boat1	UAV123	UAV123	1280x720	
boat2	UAV123	UAV123	1280x720	
boat3	UAV123	UAV123	1280x720	
boat4	UAV123	UAV123	1280x720	
boat5	UAV123	UAV123	1280x720	
wakeboard1	UAV123	UAV123	1280x720	yes
wakeboard2	UAV123	UAV123	1280x720	yes
wakeboard3	UAV123	UAV123	1280x720	yes
wakeboard4	UAV123	UAV123	1280x720	yes
wakeboard5	UAV123	UAV123	1280x720	yes
wakeboard6	UAV123	UAV123	1280x720	yes
wakeboard7	UAV123	UAV123	1280x720	yes
wakeboard8	UAV123	UAV123	1280x720	yes
wakeboard9	UAV123	UAV123	1280x720	yes
cyclist0	Online Source [32]	Myself	840x480	
cyclist1	Online Source [32]	Myself	840x480	
cyclist2	Online Source [32]	Myself	840x480	
cyclist3	Online Source [32]	Myself	840x480	
cyclist4	Online Source [32]	Myself	840x480	
cyclist5	Online Source [32]	Myself	840x480	
cyclist6	Online Source [32]	Myself	840x480	
cyclist8	Online Source [32]	Myself	840x480	
cyclist10	Online Source [32]	Myself	840x480	
cyclist11	Online Source [32]	Myself	840x480	
rowing1	Online Source [40]	Myself	1280x720	
rowing2_1	Online Source [33]	Myself	840x480	
rowing2_2	Online Source [33]	Myself	840x480	
giro1	Online Source [41]	AUTH	1920x1080	
giro2	Online Source [41]	AUTH	1920x1080	
football1_1	AUTH	AUTH	1280x720	
football1_2	AUTH	AUTH	1280x720	

Table 4: List of videos in the Sports data set showing sources and the author of the annotation work

References

[1] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg, "Discriminative scale space tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[2] R. Mur-Artal, J. Montiel, and J. Tardos, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, pp. 1147–1163, 2015.

[3] G. West, "Drone on," *Foreign Affairs*, vol. 94, pp. 90–97, 2015.

[4] H. Thorpe, "Action sports, social media, and new technologies," *Communication & Sport*, vol. 0, no. 0, p. 2167479516638125, 0.

[5] A. Perlman. 10 best drones with a camera: Top choices for aerial photography. uavcoach. [Online]. Available: http://uavcoach.com/drone-with-camera/

[6] S. Dubuisson and C. Gonzales, "A survey of datasets for visual tracking," *Machine Vision and Applications*, vol. 27, no. 1, pp. 23–52, 2016.

[7] D. A. Lisin, M. A. Mattar, M. B. Blaschko, E. G. Learned-Miller, and M. C. Benfield, "Combining local and global image features for object class recognition," in *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on.* IEEE, 2005, pp. 47–47.

[8] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel, "A survey of appearance models in visual object tracking," *ACM transactions on Intelligent Systems and Technology (TIST)*, vol. 4, no. 4, p. 58, 2013.

[9] T. Tuytelaars, K. Mikolajczyk *et al.*, "Local invariant feature detectors: a survey," *Foundations and trends® in computer graphics and vision*, vol. 3, no. 3, pp. 177–280, 2008.

[10] C. Harris and M. Stephens, "A combined corner and edge detector." in *Alvey vision conference*, vol. 15, no. 50. Citeseer, 1988, pp. 10–5244.

[11] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[12] B. Green, "Canny edge detection tutorial," *Retrieved: March*, vol. 6, p. 2005, 2002.

[13] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

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

[15] 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.

[16] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[17] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–I.

[18] H. Yang, L. Shao, F. Zheng, L. Wang, and Z. Song, "Recent advances and trends in visual tracking: A review," *Neurocomputing*, vol. 74, no. 18, pp. 3823–3831, 2011.

[19] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 25, no. 5, pp. 564–577, 2003.

[20] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via online boosting." in *Bmvc*, vol. 1, no. 5, 2006, p. 6.

[21] K. Okuma, A. Taleghani, N. d. Freitas, J. J. Little, and D. G. Lowe, "A boosted particle filter: Multitarget detection and tracking," *Computer Vision-ECCV 2004*, pp. 28–39, 2004.

[22] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, "Struck: Structured output tracking with kernels," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 2096–2109, 2016.

[23] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2544–2550.

[24] 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*, vol. 37, no. 3, pp. 583–596, 2015.

[25] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on signal processing*, vol. 50, no. 2, pp. 174–188, 2002.

[26] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 2. IEEE, 2000, pp. 142–149.

[27] L. W. Kheng, "Mean shift tracking," *Technical report, School of Computing, National University of Singapore*, 2011.

[28] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *Acm computing surveys (CSUR)*, vol. 38, no. 4, p. 13, 2006.

[29] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, “Accurate scale estimation for robust visual tracking,” in *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press, 2014.

[30] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, “Learning spatially regularized correlation filters for visual tracking,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4310–4318.

[31] M. Mueller, N. Smith, and B. Ghanem, “A benchmark and simulator for uav tracking,” in *European Conference on Computer Vision*. Springer, 2016, pp. 445–461.

[32] J. Lund. Road cycling whit dji inspire 1 filmed in 4k uhd. Youtube.

[33] john edwards. Rowing on the river thames, wandsworth, london: Drone. Youtube. [Online]. Available: https://www.youtube.com/watch?v=_-d6lgoPJts

[34] Y. Wu, J. Lim, and M.-H. Yang, “Online object tracking: A benchmark,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2411–2418.

[35] T. Vojir, J. Noskova, and J. Matas, “Robust scale-adaptive mean-shift for tracking,” *Pattern Recognition Letters*, vol. 49, pp. 250–258, 2014.

[36] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2006.

[37] R. Mur-Artal and J. D. Tardos, “Orb-slam2: an open-source slam system for monocular, stereo and rgb-d cameras,” *arXiv preprint arXiv:1610.06475*, 2016.

[38] J. Engel, J. Sturm, and D. Cremers, “Semi-dense visual odometry for a monocular camera,” in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 1449–1456.

[39] J. Engel, J. Stückler, and D. Cremers, “Large-scale direct slam with stereo cameras,” in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 1935–1942.

[40] Q. Thames. Parker cup 2016 novice 8. Youtube. [Online]. Available: https://www.youtube.com/watch?v=XdoCUQo_nCg

[41] TopCyclingH. Giro d’italia 2015 full hd 1080p. YouTube. [Online]. Available: https://www.youtube.com/watch?v=E40gav2uVhA