Feature Based Object Tracking:

A Probabilistic Approach

by

Kaleb Smith

Bachelor of Science
Electrical and Computer Engineering
College of Engineering
2016

A thesis
submitted to Florida Institute of Technology
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Computer Engineering

Melbourne, Florida
December, 2016

We the undersigned committee
hereby approve the attached thesis

Feature Based Object Tracking:
A Probabilistic Approach by Kaleb Smith

Dr. Anthony O. Smith
Assistant Professor
Electrical and Computer Engineering
Committee Chair

Dr. Adrian M. Peter / Outside Committee Member
Associate Professor
Engineering Systems
Outside Committee Member

Dr. Gerogios Anagnostopoulos
Associate Professor
Electrical and Computer Engineering
Committee Member

Dr. Samuel P. Kozaitis
Professor and Department Head
Electrical & Computer Engineering

ABSTRACT

Title:

Feature Based Object Tracking:

A Probabilistic Approach

Author:

Kaleb Smith

Major Advisor:

Dr. Anthony O. Smith

Video analysis is a rich research topic, due to the wide spectrum of applications such as surveillance, activity recognition, security, and event detection. One of the important challenges in video analysis is object tracking, which provides the ability to determine the exact location of an object of interest within each frame. Many challenges affect the efficiency of a tracking algorithm such as scene illumination change, occlusion, scaling change and determining a search window from which to track object(s). We present an integrated probabilistic model for object tracking, that combines implicit dynamic shape representations and probabilistic object modeling. We demonstrate the proposed tracking algorithm on a benchmark video tracking data set, and achieve state-of-the art results in both overlap-accuracy and speed.

Table of Contents

List of Figures

List of Tables

Acknowledgements

This work was partially supported by the National Science Foundation under Grant 1560345 and Grant 1263011. This work was also supported by graduate student research assistantship under Harris Corporation. Any opinions, findings, and conclusions or recommendations expressed in this work are those of the authors and do not necessarily reflect the views of the National Science Foundation or the Harris Corporation.

I would like to first and foremost thank my advisor, Dr. Anthony O'Neil Smith, who through this process has been a colossal help to my success thus far in my higher education experience. Through example and mentoring you have shown me what it takes to be an impact in our community. To Dr. Adrian Peter, thank you for always pushing me to know more than I already know and never settling for average. To Dr. Georgios Anagnostopoulos, thank you for your constant humor and expectations beyond our lab; it has driven me to look outside the box when solutions are not apparent. To the members of the ICE lab, thank you for putting up with my extensive social behavior and for giving me your time when I feel like acting out on it. Specifically, to Mark Moyou; you are a great example of what it takes to get through a Ph.D, through your actions you have helped me greatly through these past years, and for that I thank you.

I would like to also thank Cheryl Mitravich and Tania Watkins of the ECE department. You two have made my life as a graduate student as easy and painless as possible. I owe you both dearly for all your time and effort, thank you.

To all my friends, thank you all for always asking how I'm doing, for smiling and saying hello when you see me, and for being part of my life. To my best friend Jimmy Brennan, thank you for believing, for being interested in my work, and for your engineering attitude to help solve a problem whenever there is one. To my dear friend Kaylen Bryan, thank you for pushing me in the gym, in the class, and in life. To my brother, Elliott Bennett, thank you for your support and your drive even if you are a thousand miles way.

To my mother and father, Diana and Bill, thank you for believing in me even when you have no clue what I'm doing. Your love and support has always shown its way through my work ethic and the way I carry myself through life. To Merri and Duke, you both have excelled in what it means to be "man's best friend. You two help ease any stress or frustration I have felt throughout this process, I am incredibly in your debt and I hope your lives show how much I truly love you two. Last, but not least my beautiful fiance Julie, thank you for putting up with me, for your constant love, your constant care, and your constant support. I would not be where I am today without you. With this we are one step closer to the end goal and I know you'll be right there smiling beside me throughout the way. I love you dearly, and can not thank you enough for all you do.

Dedication

I would like to dedicate this master thesis to my late grandfather, William Earl Wilson. I've been told I look like you, I talk like you, and apparently I think like you. I wish I could share this with you and see you ponder the high points and flaws in my work. Thank you.

Chapter 1

Introduction

Tracking is a fundamental task in video sequence analysis. The resulting tracks can be used to analyze behaviors or predict the trajectories of objects in the scene, similar to work in [17]. The goal of visual tracking is to locate, track, and analyze one or more objects of interest in a video. Many important applications today are based on visual tracking such as human-computer interaction [42, 3], surveillance [26, 28], video editing [34, 13], vehicle navigation [5, 59], etc. In these applications, tracking algorithms are used to analyze video frames and extract object(s) of interest throughout the continuous sequence. Many challenges affect the efficiency of a tracking algorithm such as the change in background light, the presence of occlusions, scaling, and determining the search window for the tracked object(s).

We present a method to track the shape of a dynamic object in video. We construct a joint 2D density function that encapsulates both dynamic shape and appearance models, in which a contour of the object is used to identify precise location. This is accomplished by constructing a cooccurrence probabilistic model of the object to be tracked. Using cooccurrence was introduced by [29] as a method of

texture analysis to extract meaningful features for image classification. The image cooccurrence work was extended by [2] for color image classification, where the authors computed pairwise within and between color band joint density functions. In previous work [47] developed a generalization for visual object tracking using a multi-dimensional probabilistic model in a CAMSHIFT framework [9] which was derived from the earlier Mean Shift algorithm[14]. The approach demonstrated a robust algorithm with the capability to accurately track in complex environments, but the method has its drawbacks with respect to performance and only using red (R), green (G), blue (B) features. Computing per frame pair-wise joint probability models for the color band combinations, creating a combined probability image, along with the CAMSHIFT framework, is computationally expensive for a practical tracker. Another limiting factor in previous work is the background noise encountered when attempting to model the object given a specific search region. During this process, image noise contributes to the total probability image, making it difficult at times to find the object's accurate position and pose. Finally, in utilizing the CAMSHIFT framework, the authors were limited to only providing a rectangular bounding box for the object position. However, for many applications, such as activity recognition and object identification, it is important to have the ability to provide high fidelity object contours.

Our contribution overcomes the computational task by developing a tracker that takes advantage of general purpose graphics processing unit (GPGPU) accelerator hardware to provide real-time tracking capability. Also, by deploying an object foreground extraction technique, we mitigate image noise interference. In cases of complex object motion, relative viewpoint change, and object occlusions, our method demonstrates comparable tracking accuracy results while maintaining

real-time processing capability. We demonstrate results on the visual object tracking (VOT) standard data set. The remainder of this paper is organized as follows: Chapter 2 will provide a summary of state-of-the-art single object tracking algorithms, Chapter 3 will provide the probabilistic tracking framework, Chapter 4 will detail our research contribution, Chapter 5 will provide a snapshot of significant results, and Chapter 6 will outline some discussion points and future work.

Chapter 2

Literature Review

Video object tracking is a major field in computer and machine vision. This area of research consists of single object tracking and multiple object tracking as well as object detection and recognition. Single object tracking identifies only one object within a video frame, while multiple object tracking identifies several objects in a frame. This is best explained by example of a football game on television; you could track the football throughout the entire game (single object tracking) or track all the wide receivers on the field at once (multiple object tracking). Typically, multiple object tracking requires the object to be detected when it enters a video frame. This requires an algorithm to recognize (or classify) a new object once it enters the frame, then determine if this is the object of interest [6]. This is not the case for single object tracking, which is restricted to one.The focus of this thesis is accurate and efficient single object tracking throughout a video sequence.

The popularity of video tracking in computer vision, has led to countless attempts to overcome obstacles that affect tracking accuracy. These attempts have scattered the proposed methodologies into a few common categories. One such cat-

egory is a type of neural network and classification tracker [49, 23]. Their approach is to learn possible regions of interest (ROI), and rank them as contenders for the possible object of interest (OOI). The subset of objects, is then classified based on extracted features in order determine which ROI for the selected OOI is most prevalent [31, 1]. This category of trackers, consistently produces high overlap accuracy which is a statistical measure of how well the predicted ROI coincides with the ground truth ROI. They also achieve low miss rates, that indicates a count of how many times the predicted track completely misses the OOI [39].

A second category of tracking algorithms includes per frame motion and geometry feature extraction [32, 4, 51]. This class of trackers has low computational cost, but does not perform well in terms of accuracy and miss rate. The fundamental idea of motion based trackers, is to estimate the directional motion of an OOI from frame to frame. This method is extremely successful when the capture device is a static camera, however when the camera platform is dynamic it introduces an challenging component.The moving object can seem to have zero motion if moving with the same relative velocity as the camera [66].

Geometric trackers extract contours that define the shape of the OOI. This category of trackers, makes an attempt to separate foreground from background in order to isolate the OOI from noise [16]. Foreground is not always partitioned correctly and the result is that concluded contours are not always indicative of the OOI. Typically, motion and geometric based trackers are not standalone due to their poor tracking. In most cases they are utilized to enhance other algorithms.

Due to the amount of unique information that can be extracted from a video frame, feature tracking has become the most popular approach for single object tracking. A similar set of features are extracted are calculated across different

regions. A similarity metric like Euclidean distance (linear feature space), is used to compare regions with the OOI and identify the closest match. Some common features that have shown success under various conditions are: textures, corners, edges, energy, SIFT (scale invariant feature transform), SURF (speed up robust features), or ORB (oriented fast and rotated binary robust independent elementary features) [67, 58, 43, 30, 11, 64, 15]. Some features have the additional property of being invariant to particular fluctuations in the scene. This makes them more robust to obstacles during tracking. For instance, SIFT, SURF, and ORB are scale invariant features, meaning they are not impacted by a change in camera zoom. These robust features are typically used in combination to create high dimensional feature vectors that describe the OOI. The drawback that these approaches present is the high computational cost and memory utilization to store all the features per pixel. To combat this, we propose to compute a covariance matrix from features associated with the OOI [55]. The covariance matrix framework allows different features to be fused into one matrix descriptor that is dependent on dimensionality of the feature vector. This covariance matrix can represent the features used to describe an OOI/ROI, and has demonstrated to be as effective in tracking accuracy and miss rate as pixel wise feature vectors [60]. Where the covariance matrix framework excels, is in its ability to be the blender, so to speak, which combines multiple robust features.

In this thesis we focus on a class of single object probabilistic trackers that is different from those previously mentioned. To be more specific they rely on statistical probability values which are obtained about the object/region of interest and then used to track throughout the video sequence. Just as the other classes, there are many probabilistic tracking algorithms in the field today. Kalman Filter

tracking [25, 10, 46] assumes the video model is Gauss-Linear and is able to predict where the object will be in the next frame using a Gaussian distribution. However, most video tracking problems are not Gauss-Linear and have non-Gaussian noise, non-linear observations, and non-linear dynamic movement causing Kalman Filter tracking to be inapplicable in these situations. Several attempts were made to make Kalman Filtering more practical for video tracking. The authors of [36] used local linearization of dynamics to approximate the non-linear dynamics that is present due to motion in video typically. In their Extended Kalman Filter (EKF), the authors of [63] drew a number of samples to simulate motion dynamics and then compute a mean and covariance of new points in their Unscented Kalman Filter (UKF) [8]. Even these attempts cannot solve the underlying issue that the probability posteriors for video tracking are never Gaussian or unimodal.

Particle filter tracking is another probabilistic tracking method where the algorithm infers information by approximating distributions of different sets of points (particles), and does this without making irrational assumptions of the video model [33]. There are some disadvantages for this method as well. For instance, the ideal number of particles for a good distribution is dependent on how much computation is acceptable. Another option is to change the video motion model and take advantage of more domain knowledge in order to produce better distributions. Other extensions to particle filtering include the Rao-Blackwellized Particle Filter, which attempts to reduce the posterior entropy by adjusting state variables [38] and the Auxiliary Particle Filter which tries to look ahead at observations to continuously build a better distribution. There has also been some research done for tracking body motion on camera using an annealed particle filter [24].

In the computer vision and machine learning community, cooccurrence his-

togram tracking has gained momentum. This method consists of computing neighborhood relations between pairs of pixels within a predefined radial distance. These pairs of occurrences are summed and stored in a two dimensional histogram [30] that is then used to form a probability density function [18]. The probabilities from the cooccurrence matrix are then back projected in a way that enables the ability to track an object of interest. The question here is that once you have obtained the back projection of probabilities to represent your object, how do you maintain a track on the object, and what color space is ideal to develop the most accurate back projection? To answer the first of these, there are several methods of tracking on a probability back projection. The most well known is the Mean Shift algorithm by [27]. Mean Shift is a mode seeking algorithm that is searching for the peak in the probability density function. This mode seeking algorithm helps to track non-rigid objects that can occasionally be modeled with cooccurrence histograms. Mean Shift process begins at an initial location, computes the mean within a radial distance, and then moves to the new mean before starting the computation again. The process repeats until the mean does not move within a convergence criteria, which indicates you have reached a maxima. That location is then used as centroid of your updated ROI [14, 20]. The initially formulation of this algorithm was slow due to its mode seeking nature, which checks different mean points as centroids for convergence. However, because it has been adapted to real time processing [21], it is still being used to support tracking needs [67, 19, 65, 35]. However, this algorithm does not, adapt to different bounding box search window sizes, which could affect tracking when there is video zoom in the scene.

The gaps in the Mean Shift algorithm led to one of the most popular probabilistic tracking algorithms to date: Continuously Adaptive Mean Shift (CAMSHIFT)

by the authors of [9]. This algorithm takes the same approach as Mean Shift, however, when the centroid is found, the search window or bounding box is calculated with respect to the zeroth moment [7] of the image. This allows the search window to adapt and compensate for scale changing objects of interest due to zoom or camera motion.

Originally, CAMSHIFT was made to track human faces, leading to the discussion of our second question mentioned above - what is the best color space on which to calculate cooccurrence histograms? For human faces and flesh, the HSV (hue-saturation-and value) color space performs best [67, 22]; however, for other objects the HSV color space performs poorly in tracking applications. The authors of [12] found that using the red-green-blue(RGB) color space performed well on objects with drastic color difference, however, the cooccurrence was percalculated using several different aspects of the object of interest and combining them in order to compute a more robust cooccurrence matrix. This methodology is impractical to video tracking since real time applications are desired, but even in forensic video tracking there are instances where no information is available for the object of interest. It was in [48] that the authors produced a new method of color histogram tracking, where they formed tuple pairs of independent RGB color space bands and fused into an estimated total probability density function. This algorithm is the foundation of our proposed approach. As apposed to RGB, we discuss the use of different features to build a feature image, for which we will create a back projection that is then feed into CAMSHIFT.

Chapter 3

Overview of Tracking Framework

In our research, our goal is to develop a real time single object tracker. As described in the previous sections, most approaches utilize either geometry, classification, or a single set of derived features. We propose a framework that will allow us to define a dynamic pool of per frame features that will track our object of interest. We also implement our solution in real time by taking advantage of the latest GPGPU high performance computing technology.

3.1 Initial Processing

We begin by defining some notation in order to describe the framework for the system. Let T_k be the target information estimated by the tracker at frame I_k and defined as

$$T_k = \left(x^u_{k,}, y^u_k, x^l_{k,}, y^l_k, ROI_k\right)$$

Figure 3.1: Geometric figure of how α is added to the initial target to get the ROI_{init} [56].

where $k = 1, \ldots K$. K is the number of frames in the video sequence. Here (x_k^u, y_k^u) and $\left(x_k^l, y_k^l\right)$ define the upper left and lower right point positions for the bounding box around the target. ROI_k is the search region of interest on the frame and presented in the form of a bounding box.

The tracking algorithm begins with the user initialization of the target T_{init} , or some indication of the target initial bounding box (x_{init}^u, y_{init}^u) and $\left(x_{init}^l, y_{init}^l\right)$. In Figure (3.1) we see the initial region of interest ROI_{init} is estimated by adding an offset α, which controls the amount of background scene $ROI_{init} = (x_{init} \pm \alpha, y_{init} \pm \alpha)$. Intuitively, the region of interest is a selected subset of pixels in the frame for the purpose of defining a controlled search space. This technique is used in many applications in order to improve processing performance by localizing the area for which the estimated target may exist.

Figure 3.2: Initial target, and target after the background is removed.

In the second stage of pre-processing we perform foreground estimation and isolate T_{init} by removing background pixels that do not contribute to the target region. Figure (3.2) illustrates our use of the *GrabCut* algorithm by [57]. The algorithm employed to perform foreground extraction with minimal user interaction and a single video frame.

The *GrabCut* algorithm works by initializing a bounding rectangle around the foreground region. A Gaussian Mixture Model (GMM) is used to model the foreground and the background independently. The algorithm interactively segments

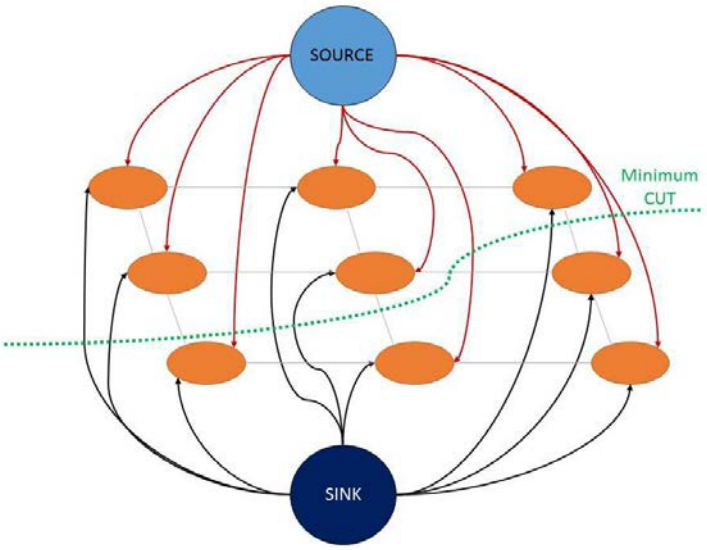

Figure 3.3: Illustration of "GrabCut" algorithm

the region to achieve the desired results. In Figure (3.3) we show what occurs during each iteration, where the GMM learns and creates a new pixel distribution. An unlabeled pixel is labeled either foreground or background depending on its relationship with neighboring labeled pixels in terms of color statistics, similar to a traditional clustering algorithm. A graph G, where the vertices V in the graph are pixels, and edges E , is built from the pixel distribution. Additionally, a **Source node** and a **Sink node** are added to the graph. Every foreground pixel is connected to the Source node, and every background pixel is connected to the Sink node. The edge weights of pixels connected to the source/sink node are defined by the probability of a pixel being foreground/background respectively. The edge weights that connect two pixels are defined by the edge information or pixel color similarity. If there is a large difference in pixel color, the edge between them is assigned a low weight.

Once all edges are weighted, a min-cut algorithm similar to the work described

in [37] is used to segment the graph. Traditionally, the min-cut problem can be stated as follows: Given a graph $G = (V, E)$ and weights w_e for each edge $e \in E$, compute a minimum cut in G defined as the cut of least weight in the graph. In our case this means that the algorithm cuts the graph into two, separating source node and sink node with minimum cost. The cost is the sum of all weights of the edges that are cut. Ideally, after the cut, all pixels connected to the Source node become foreground and all pixels connected to the Sink node are background.

In our final pre-processing stage we generate a binary map of T_{init} by a process of intensity level slicing. In a number of applications binary images can be used as the input to algorithms that perform useful tasks. A binary image B will be obtained from the bounding box region of the color video frame T_{init} . The operation selects a subset of the frame as pixels of interest (intensity=1) in an image analysis task, while leaving the remaining as background pixels (intensity=0) to be ignored. The initial region from the color video frame is first converted to gray scale (intensity level 0-255). The selection operation can be as simple as the thresholding operator that chooses pixels in a certain subspace of gray scale levels.

The pixels of the binary image B are $0's$ and $1's$; the $1's$ will be used to denote foreground pixels and the $0's$ background pixels. The term $B[x, y]$ denotes the value of the pixel located at row x, column y of the image.

This binary map $B[x, y]$ will be used as a mask and enable the algorithm to calculate the cooccurrence matrix with only foreground pixels from the object of interest. This will greatly eliminate background noise that would contribute to the probability required for our tracking algorithm.

Table 3.1: The eight neighbor directions for pixel calculation.

(x-1,y+1)	(x,y+1)	(x+1,y+1)
(x-1,y)	(x,y)	(x+1,y)
(x-1,y-1)	(x,y-1)	(x+1,y-1)

3.2 Probabilistic Framework

Our study of using the cooccurrence framework for tracking is not unique. In fact, cooccurrence texture analysis is a branch of image processing that has been studied for more than 40 years. For a review of fundamental work, see [30] and [61]. The original work was later generalized for color images by [2], and then applied to video tracking in [48]. Our goal in this research is to explore a generalization of the framework in order to utilize a dynamic set of features.

We use the classic gray scale image cooccurrence matrix formation to describe the formulation. Let I_k be a gray scale video frame of L gray levels. Let $s = (x, y)$ be the position of a pixel in I_k and $\mathbf{t} = (\Delta x, \Delta y)$ be a translation vector. The cooccurrence matrix $M_{\mathbf{t}}$ is a $L \times L$ matrix whose $(i, j)^{th}$ element is the number of pixel pairs separated by $\mathbf{t}$ that are equal to the gray levels $I_k(s) = i$, and $I_k(s + \mathbf{t}) = j$. The choice of the translation vector $\mathbf{t}$ is the distance of one pixel in eight neighbor directions for each pixel.

The eight matrices are summed to obtain a rotation-invariant symmetric matrix M where $M_{\mathbf{t}}(i, j) = M_{-\mathbf{t}}(j, i)$. More formally written every $M_{\mathbf{t}}$ element is defined by

$$M_{\mathbf{t}}(i, j) = card\left\{(s, s + \mathbf{t}) \in \mathbf{R}^2 | \, I[s] = i, \, I[s + \mathbf{t}] = j\right\}. \tag{3.1}$$

From the basic gray scale single band example, we wish to describe how to extend the method of cooccurrence to a multi-band image, i.e. images encoded

with n bands. Let b_i be defined as a single band of the k^{th} frame, where there are n possible coded bands in a video sequence and $i = 1, 2, u, v, \ldots, n$. $B = (b_u \rightarrow b_v)$ and, $B^{-1} = (b_v \rightarrow b_u)$ are the coupling of bands b_u and b_v, and similarly, $\mathbf{t} = (\Delta x, \Delta y)$ is a translation vector. B indicates that in the coupling pixels defined by $\mathbf{t}$, the first belongs to band b_u and the second to b_v. The generalized cooccurrence matrices are

$$M_{\mathbf{t},B}\,(i,j) = card\left\{(s, s+\mathbf{t}) \in \mathbf{R}^2 |\, b_u\,[s] = i,\, b_v\,[s+\mathbf{t}] = j\right\} \tag{3.2}$$

with one matrix per B coupling of bands. Again, the translation of $\mathbf{t} = 1$ pixel in the eight neighboring directions to obtain M_B, the summation of the eight matrices. The symmetric property remains true, so $M_B = M_{B^{-1}}$.

Once we apply the eight neighbor scheme, this encodes the pixel texture information. Executed on a standard 3-band (b_n) where $n = 3$, that are typically represented as the Red(R), Green(G), Blue(B) color bands, it produces equal size $d \times d$ cooccurrence matrices, that contain rotation invariant features. So, to generate more robust feature information several same band, and cross band pairwise combinations are used to yield cooccurrence matrices. Using the same RGB color band example, this would be equivalent to cooccurrence matrices for (R,R), (G,G), (B,B), (R,B), (R,G), and (G,B).

The strength of the cooccurrence matrix formulation is that after being normalized, they can be interpreted as independent joint density functions for each band combination. For a given pixel intensity value, we can lookup the corresponding probability from the joint density function, then back project to obtain the probability image. The probability density function of each cooccurrence matrix

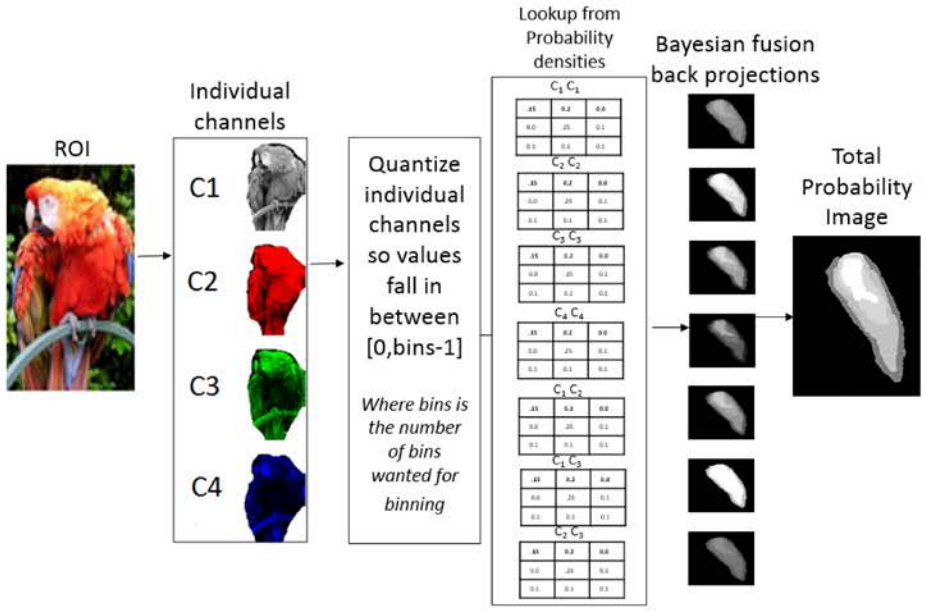

Figure 3.4: Initial process of ROI

can simply be obtained by dividing each element by the sum of the cooccurrence matrix.

$$P(M_{band}) = \frac{M_{band}(i,j)}{\sum_{i=1}^{bins}\sum_{j=1}^{bins} M_{bands}(i,j)} \tag{3.3}$$

Intuitively, this describes the probability of a channel's single element value being neighbors with its eight closest pixels. The entire initial processing sequence for an ROI (RGB color space in the example) is shown in Figure (3.4)

3.3 Video Sequence Processing

After the initial processing is done, the task of tracking the object of interest needs to be performed. The tracker will start tracking the OOI on the first frame given $T_1 = (x_1^u, y_1^u, x_{1,}^l y_1^l, ROI_1)$ to start the tracking sequence. The tracker will continue estimating ROI_k for $k = 2, ...K$. Of course, if the track is lost on the object of

0	0	1	6	3	8	4	4	4	0
0	2	3	3	3	3	1	1	1	0
0	2	3	3	3	3	8	8	8	0
0	2	4	5	5	5	5	8	8	0
0	7	4	5	5	5	5	5	0	0
0	7	8	5	5	5	5	3	1	0
0	4	4	5	5	5	5	3	4	0
0	4	4	4	4	8	4	4	1	0
0	4	8	8	8	8	2	3	1	0
0	0	0	0	0	0	0	0	0	0

$\rightarrow$

0	0	0	2	1	2	1	1	1	0
0	0	1	1	1	1	0	0	0	0
0	0	1	1	1	1	2	2	2	0
0	0	1	1	1	1	1	2	2	0
0	2	1	1	1	1	1	1	0	0
0	2	2	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	2	1	1	0	0
0	1	2	2	2	2	0	1	0	0
0	0	0	0	0	0	0	0	0	0

Figure 3.5: Left: 10×10 ROI, Right: quantized new 10×10 ROI

interest, the real time tracking is null and ineffective. However, the algorithm does not know this and will continue to perform its tasks on the ROI that is reported.

The first step is similar to that in pre-processing. Continuing the multi-band image example where frame k is composed of b_n bands, and in our case $n = 3$ which is the standard RGB image. The ROI must be split into independent color band channels and also, converted to gray scale. These channels are then quantized so each pixel's value falls in the interval $[0, d-1]$. This binning technique makes calculations more efficient, as well as polls like pixel intensities.

Since our cooccurrence matrix stores the values in a $d \times d$ matrix depicting the bins' occurrences in the ROI, when converted to $P(M_{band})$ we now have a probability of the specific bins' occurrence happening in the ROI. This concept is what will be used during each frame to look up the pixel's bin occurrence. For example, say we have an ROI that is $10 \times 10 \times 3$ and we only want to look at one band of this ROI. Let $d = 3$, now quantize the ROI to fall in between $[0, 2]$ seen in Figure (3.5).

$P(M_{band})$ is the total probability density at the locations of bin occurrences calculated from our initial processing from the first frame. This total probability

0.15	0.05	0.05
0.05	0.3	0.15
0.05	0.15	0.05

$\Rightarrow$

0.1	0.1	0.1	0.05	0.1	0.05	0.1	0.1	0.1	0.1
0.1	0.15	0.2	0.3	0.15	0.3	0.15	0.1	0.1	0.1
0.1	0.15	0.2	0.5	0.5	0.3	0.3	0.15	0.15	0.1
0.1	0.2	0.3	0.5	0.5	0.5	0.5	0.2	0.15	0.1
0.1	0.05	0.05	0.5	0.5	0.5	0.5	0.3	0.15	0.1
0.1	0.05	0.3	0.5	0.5	0.5	0.5	0.3	0.1	0.1
0.1	0.05	0.3	0.5	0.5	0.5	0.5	0.3	0.2	0.1
0.1	0.05	0.3	0.3	0.3	0.5	0.3	0.2	0.2	0.1
0.1	0.05	0.05	0.05	0.05	0.1	0.1	0.1	0.15	0.15
0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.15	0.15

Figure 3.6: Left: $P(M_{band})$, Right: Back projection from $P(M_{band})$ to ROI

will not change per frame, therefore it shows the importance of collecting the cooccurrence of the actual OOI. The total probability is a look up, that is then back projection and the pixel location is filled in with the associated value. The back projection process is described in detail in [48], intuitively for every pixel (x, y), the probability of the intensity value $I_k(x, y)$ existing in the ROI is

$$P\left((x,y)\,|I\left(x^c,y^c\right)\,,\,I_{1:8}^N\right)=\prod_{t=1}^{T}P_t\left((x,y)\,|I\left(x^c,y^c\right)\,,\,I_{1:8}^N\right) \tag{3.4}$$

where t is the index of the cooccurrence matrix, $I\left(x^c, y^c\right)$ is the center pixel, and $I_{1:8}^N(x, y)$ is the eight neighbor directions. This can be seen in Figure (3.6).

The process described in the example is performed for all same band and cross band pairwise combinations; tor this example, this would be done for (Gr,Gr), (R,R), (G,G), (B,B), (R,B), (R,G), and (G,B) and captured in a probability back projection for that particular pairwise combination. As seen in Figure (3.6) there is a high concentration of high probability in the middle of our ROI. This emphasis occurs in all band combination back projections, then once fused will highlight our object of interest. It is important to note that the back projection will not sum

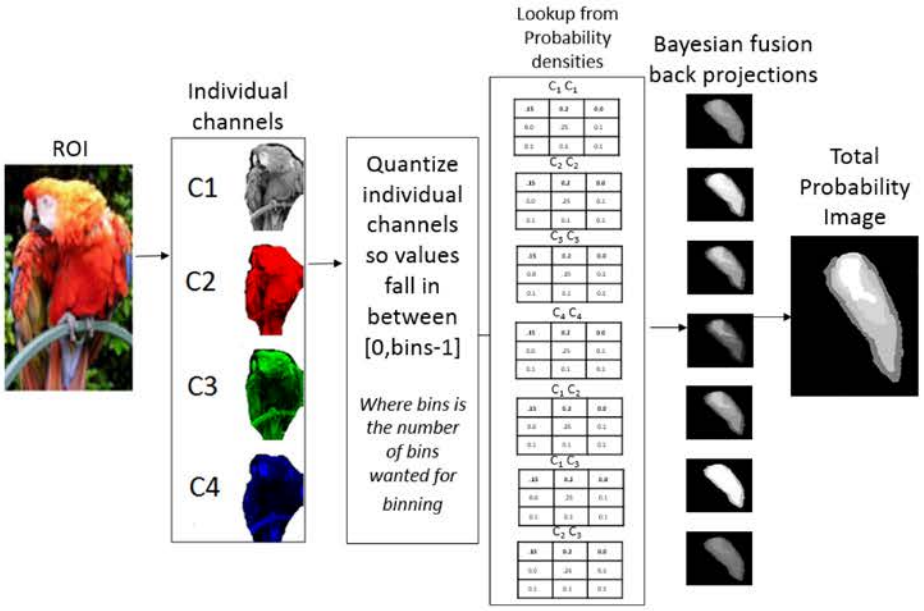

Figure 3.7: Real time video process

to one like a probability density function, since in essence, it is not a probability density, but rather a good description of how the pixel bins occur in the ROI. A complete depiction of the video processing steps can be seen in Figure (3.7).

These seven back projections are then summed for a total probability back projection image $P(I_{total})$. The back projection image is used in conjunction with the CAMSHIFT algorithm in order to converge on the maximum probability. CAMSHIFT then produces an adaptive bounding box around the OOI which is now the predicted ROI. The method is repeated for each frame, and the ideal ROI will encapsulate the OOI throughout the entire video sequence.

Chapter 4

A Probability Feature Tracker

4.1 Covariance Feature Tracking

Within the process of probabilistic tracking, an occurrence matrix (or two-dimensional histogram) is formed from pixel intensities and the neighboring pixels. The classic approach performs this method with either of two different three-dimensional color spaces, Red-Green-Blue (RGB) or Hue-Saturation-Value (HSV). The method we propose overcomes the obstacles of processing with a single one of these color spaces.

Intuitively, probability tracking can involve processing with more than just pixel intensities and its relative relation to its neighbors. Instead we propose using extracted features to define a feature space as the primary attribute for processing. The algorithm will adhere to the framework that was described in Section (3.2). In this case instead of RGB pixel intensities for the cooccurrence calculation, we employ extracted vectorized features per pixel location. This method will greatly increase a track on an object, and is invariant to size, shape, color, or change in

lighting conditions.

The per pixel covariance matrix [60] encapsulates the variation of dimension to dimensions relationship. For example, a three dimensional $\mathbf{R}^3$ data set would produce a 3×3 covariance matrix. Along the diagonal is a variance which is a measure of variation within the first dimension and the off diagonal terms are the covariances that measure across dimension variation. A positive an element in the covariance matrix, implies that the dimensions compared decrease or increase together, alternatively a negative element means the opposite. A zero element indicates that the two dimensions are completely independent of each other.

$$Cov(X, Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{n - 1} \tag{4.1}$$

To compute our local covariance matrix we use the eight neighborhood scheme seen in Table (3.1). The value at the center pixel locations will denote as $[C_1 C_2 C_3]$ for the three channel example, we calculate the covariance matrix about the eight neighbors of the center location. The next step is to produce a feature vector that is the result of a vectorized covariance matrix. This becomes the new representation for the center pixel Figure (4.1).

For our example, there are only three channels used to compose a single covariance matrix. In reality, the feature vector can be extended to some $\mathbf{R}^D$ dimensions. This allows us to generalize each pixel location with more than just RGB color pixel values, but extend to a concatenated HSV, HLS, Luv color spaces, SIFT, SURF, ORB features, texture and any number of features. By localizing the covariance matrix, we can fuse all of these D dimensional features per pixel in a more robust manner. This inevitably creates a more dynamic, robust tracker when performing a back projection for the OOI.

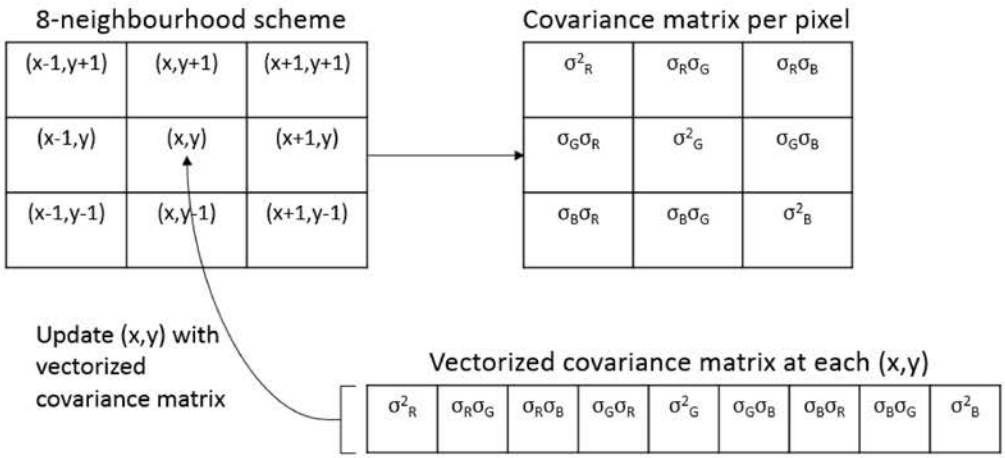

Figure 4.1: Covariance Feature depiction for feature image creation. Starts in top left and works clockwise per pixel

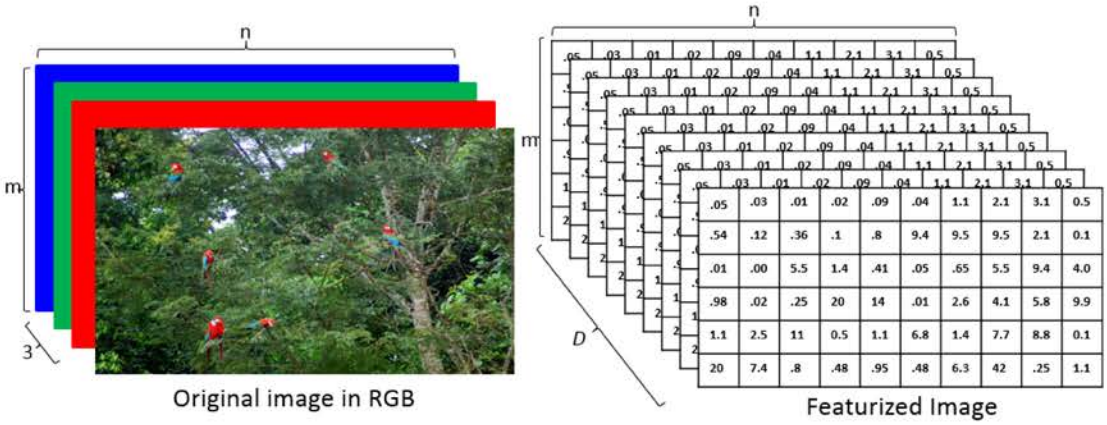

Figure 4.2: Mapping a RGB image to a D dimensional feature image

The generalization seen in Figure (4.2) is an extension to the algorithm framework presented in Section (3) for calculating cooccurrence matrices to produce back projections for histogram tracking. The feature cooccurrence matrices should develop a rigorous more robust tracking model that is invariant to many of the obstacles apparent to video tracking. Using the local per pixel covariance matrix assist in fusing different features of our OOI and enable us to better characterize the object of interest.

4.2 Performance Improvement Implementation

Performing these task on each frame, makes it apparent that this approach is computationally expensive. All the calculations performed on a frame are per pixel. Then with any increase in the ROI size the cost for calculating cooccurrence matrices, probability densities, and back projections become even greater. Not to mention that these calculations per pixel are done for seven pairwise combinations in order to compute the total probability back projection. If implemented in a traditional serial framework, this would not practical for real time applications for video streams or even playback video. Due to the advancement in high performance scientific computing hardware like the GPGPU, we were able implement the algorithm, deeming it capable of real time tracking.

Each pixel wise calculation (cooccurrence, probability density, and back projection) has been parallelized in order to perform thousands of pixel calculations simultaneously. This performance implementation is done for our initial processing, however this did not provide the most speedup. Our implementation performance benefited the most from parallelizing the frame pixel processing. This where the

look up for each pixel pairwise probability density takes place. The algorithm performs this action for all pixels at once. Furthermore, since there are seven different combinations, this per pixel parallelization is done for each combination.

With our D dimensional feature vector, there is an associated number of calculations that need to be computed on the initial ROI. Also, there are $D(D-1)$ different combinations of cooccurrence matrices that are computed in the initial processing. Part of our contribution is algorithm processing performance improvement. The speed up is achieved in the per pixel calculations. Utilizing NVIDIA GPU's [50], we reached near real time speeds.

Resulting GPU implementations have performance gains that are staggering when compared to serial processing. If the proposed framework were implemented in serial, times of approximately < 1 frame per second (fps) have been measured. Contrary to this our GPU implementation achieves measured gains of approximately $>$ 40fps. The speed up is a result of parallelization to eliminate known bottlenecks.

Chapter 5

Experiments and Results

Our algorithm was tested on various videos for single object tracking. The videos present several challenging scenes for single object tracking with obstacles that cause difficulties for tracking algorithms in the machine learning community.

5.1 Video Data Set

The data set that was used for our experiments was the Visual Object Tracking (VOT) 2016 video data set. VOT 2016 consist of sixty short video sequences which are used as a benchmark in single object tracking. The challenges in the data set include: are:

- ***Illumination change*** looks at the object's pixel color intensity and describes how it varies from the first frame. This obstacle is sensitive to changes in light. In a video this can make a dark colored object appear bright and vibrant, which causes it to lose color feature properties.

- ***Object size change*** is the sum of average local size changes. Here, local size is the mean difference between bounding box in frame k and the past fifteen frames. The size change is typically the result of camera zoom or the object moving away/towards the camera. Size change is a tracking challenge due to the object's color and size features changing rapidly and usually it is too much or too little information for the tracker to hold.

- ***Object motion*** is the mean difference between the centroid of the OOI between consecutive frames. This task is simple when an object moves in a constant linear matter; however, this isn't always the case.

- ***Clutter*** is looking at the background of the ROI and a larger region enclosing the ROI. If the compared backgrounds are similar, then the clutter is said to be high, if dissimilar then there is little clutter. Clutter can exhaust a tracker's ability to keep a track from the amount of noise caused by the similar backgrounds.

- ***Camera motion*** is similar to object motion, but focuses on the mean focal point of the camera throughout consecutive frames. This can give the illusion of movement in the scene, then if the object is actually moving the motion is enhanced and becomes chaotic that tracking is virtually impossible.

- ***Blur*** is measured by Bayes-spectral-entropy camera focus measure [40] and tries to describe videos which are low resolution or videos with blurred imaging. This can make objects appear nothing like they really look, or completely blend into the background.

- ***Aspect-ratio change*** defines the average per-frame aspect ratio changes.

It changes at frame k and is calculated as the ratio of width and height in frame k divided by the ratio of the bounding aspect ratio of frame $k = 1$. This challenge makes objects of interest seem to be changing size in the video and causes features (color and object) to change with it.

- ***Object color change*** is similar to illumination change, however the focus in only on hue of the bounding box region. The hue consists of no illumination changes, just raw color of the object. Color changes in a video change everything for trackers which rely heavily on color and texture features.

- ***Deformation*** partitions images into smaller grids (8×8) and then computes the sum of square differences of the mean pixel intensities over the grids in frame k and the first frame. Deformation is similar to aspect-ratio change and produces similar challenges to trackers.

- ***Scene complexity*** takes into account the amount of statistical entropy in the frames and is calculated as $e = \sum_{i=0}^{255} b_i log(b_i)$ where b_i is the number of pixels equal to i. The complexity of a scene is taken into account how much action occurs in the scene, the "busier" the scene, as in the motion, objects, and background, the more complex it is. Complexity is an issue to trackers because the more complex the scene, the more information the tracker must be able to differentiate from the object of interest.

- ***Absolute motion*** looks at the centroid of the ROI between frames. This is similar to the object motion except using the entire bounding box instead of the object of interest. This attribute will consider the object motion obstacle as well as the camera motion obstacle.

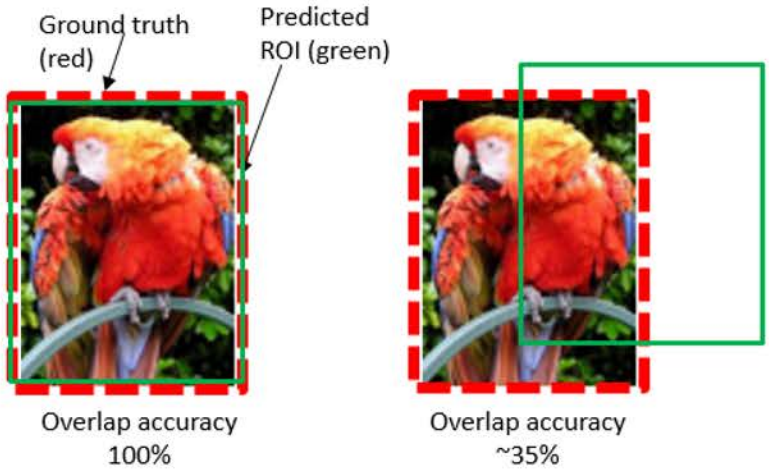

Figure 5.1: Overlap Accuracy

These obstacles occur randomly in various sources of video that include surveillance or aerial and must be overcome by trackers in order to perform well in practice.

5.2 Testing Procedure

Each video's object of interest is bounded with a ground truth box and follows the OOI throughout the entirety of the video sequence. The tracking algorithm will produce a ROI which will attempt to be as similar as possible to the ground truth in order to achieve a 100% overlap accuracy. This accuracy is one of the measures used to rank the tracker in terms of how well it can separate the OOI out of a scene. A generic depiction of overlap accuracy is shown in Figure (5.1) This prevents a tracker from using the entire frame as the ROI and keeping the track for the entire video sequence.

The other evaluation measures are ranking measure, failure rate. Failure rate is how often the predicted track comes completely off the OOI. If a failure happens

on a frame, the ROI is reinitialized as the ground truth and the track restarts. The ideal track results is to achieve a zero failure rate, meaning the predicted ROI stayed on the ground truth OOI the duration of the video sequence.

VOT 2015's toolkit has sixty video sequences which it runs for three-fifteen repetitions per video during the experimental procedure. The total output is given in an average accuracy per video, average number of frames failed per video, and the average speed at which the algorithm performed [39].

5.3 Experiments

5.3.1 Initial Experiments

Before the algorithm was placed into competition with other state of the art trackers, fine tuning needed to be done to evaluate the performance. A few experiments needed to be done to fine tune parameters. For instance, the number of bins needed for the algorithm to cluster the ROI is critical and impacts the results dramatically. This gave us an opportunity to test our covariance features in our framework with actual images. We looked at several different ROIs from the VOT data set, and compared them to the computed back projection sent to CAMSHIFT. The experimental results can be seen in Figure (5.2).

An analysis of the results, revealed we had some instabilities with our covariance features as our only producer of back projections. We can see from the ROIs that have large contrast in color variation, our algorithm using covariance features produces complete and clean back projections in each of the three bin quantities chosen. However, in scenes where the ROI is similar in color to the background, the covariance feature back projections are insufficient. While we hope to see a

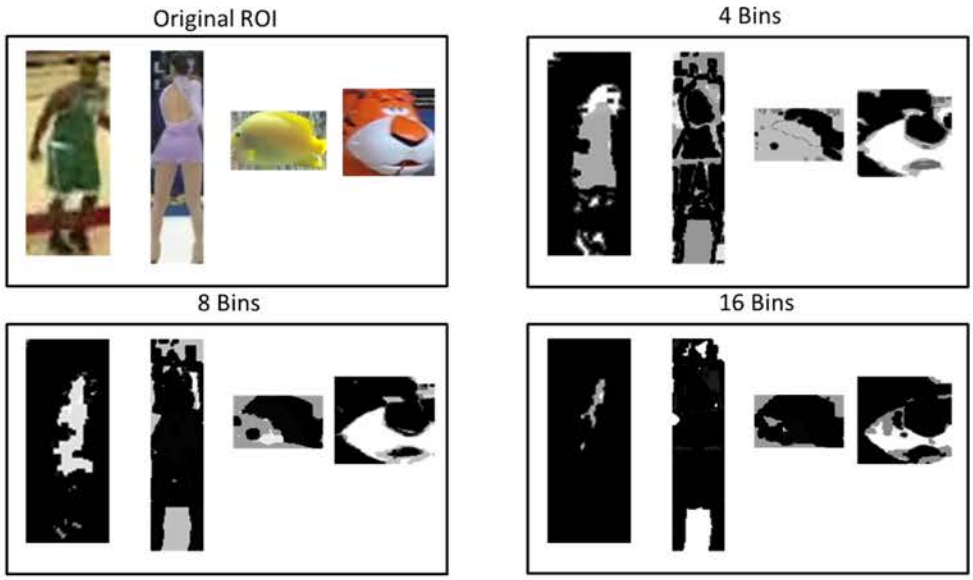

Figure 5.2: Experimental results of back projections on covariance features only

highlighted region where our OOI is located, we see emptiness, which means there is a better chance of simply guessing the OOI's location in the ROI. We can also see the importance of the number of bins needed to produce a good back projection. However, more tests are needed before deciding the optimal bin number.

Realizing that our back projections need improvement from just our covariance features, we decided to perform the experiment as seen in Figure (5.2). Instead, we used RGB only back projections, a pairwise combination with the RGB pixel intensities and the covariance features when calculating cooccurrence matrices, and a total back projection using a linear combination of all three back projections listed above. These results were more promising in our move forward and can be seen in Figure (5.3).

Covariance features cannot compete alone to produce ideal back projections. It is interesting to see that RGB back projections perform better on the ROIs where covariance features look poor. This then leads to the pairwise combination, which

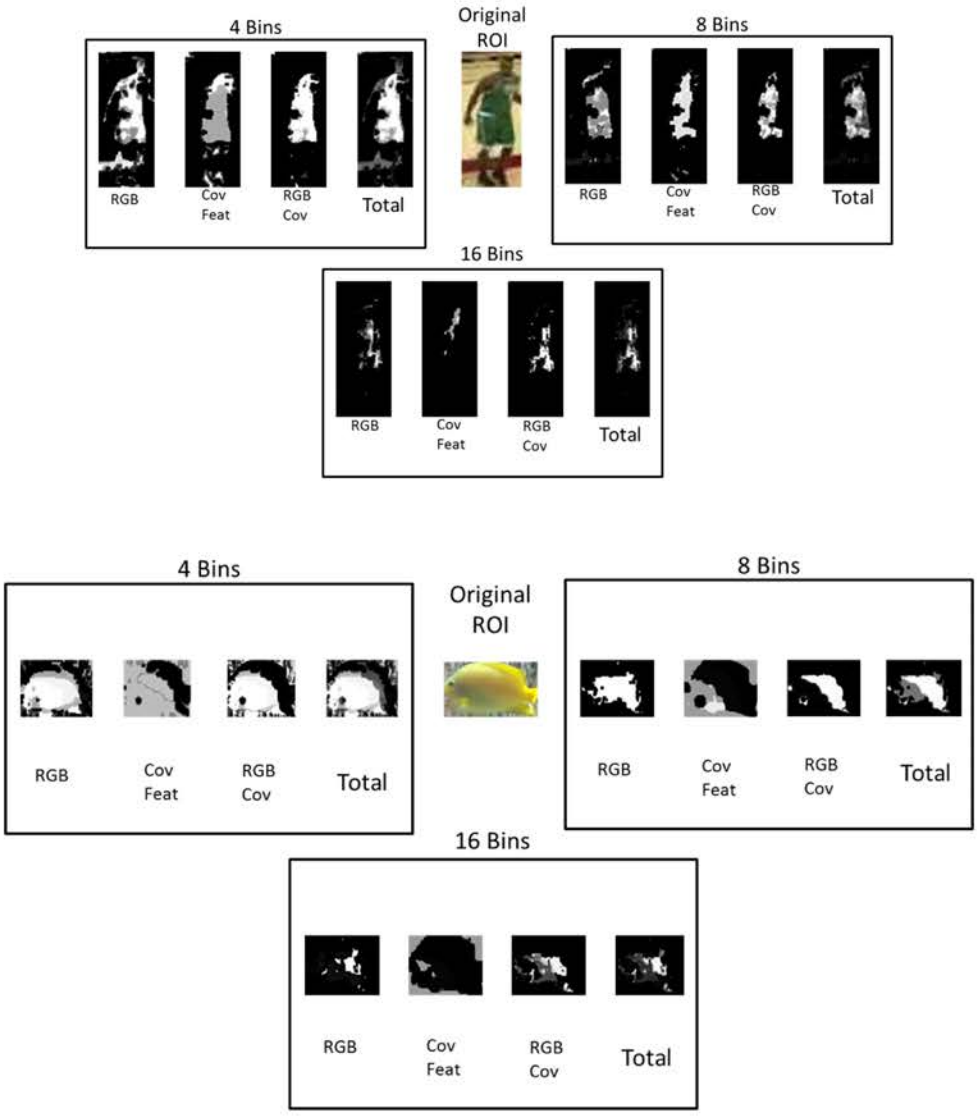

Figure 5.3: Experimental results on RGB, covariance features, RGB-Cov pairwise combinations, and total back projections.

Table 5.1: VOT 2015 experiment results: 4 bins, 8 bins, 16 bins, 32 bins

# of bins	Avg Accuracy (%)	Avg Failures (frames)
4	32.6	11.24
8	30.4	8.69
16	34	6.04
32	27.5	7.56

shows strong performance. The pairwise combinations take a covariance feature channel, and calculates cooccurrence matrices with RGB channels. This back projection performs better than singular combinations. The total back projection is a linear combination between the three independent back projections. Visually, this looks to be the strongest method, and it not only produces clean projections with less noise than the others, but it also produces strong contours around the OOI.

The total back projection is the method used to compare against competition trackers. The smaller number of bins is optimal for better back projections. This can be seen for the basketball ROI in Figure (5.3) the whole player is highlighted in the four bin selection, but there is some noise surrounding the OOI. In the fish image, the same is apparent, leading to the conclusion that if back projections visually look better then they could produce better accuracy results.

Using the total back projection explained in our earlier experiment, we ran each bin number in the VOT experiments. Table (5.1) shows the results of the experiment. The accuracy and failure rates were both more ideal with the bin number of 16. With the trend of the failures decreasing with higher number of bins, testing 32 bins was a must. This did not prove to be better, however , and the optimum for failure misses and accuracy was 16 bins. The lower number of bins created more background noise throughout the video and was the reason failure

rates were higher. Then, looking at 32 bins, it shows that not enough information can be gathered about the ROI when having too high a bin number to compress the channel values down.

Therefore, going into the evaluation part of our experiments we used 16 bins and our total probability back projection in conjunction with CAMSHIFT.

5.3.2 VOT 2015 Competition Experiment

Using the testing procedure described in Section (5.2) we ran our proposed algorithm through the VOT 2015 experiments and compared it to eight state of the art trackers that follow different classes of tracking techniques. These trackers had entered the VOT 2015 challenge and their results are available for comparison in the VOT toolkit used for ranking and comparison. These trackers include:

- ***Multi-Domain Convolutional Neural Network Tracker (MDNet)*** falls into the neural network and classification tracker class. MDNet pre-trains the tracker using a convolutional neural network (CNN) on the OOI surrounded by the ground truth provided by the data set. These stored features representing the OOI are then stored to in the CNN. Then the tracker evaluates the candidate ROIs in the scene surrounding where the initial ROI was in order to produce a maximum score for the candidate ROI closest to the original ground truth [49].

- ***Structure Tracker with the Robust Kalman Filter (RobStruck)*** another type of classification tracker, where features are not learned on a neural network but used in deciding the best candidate region for the ROI. Based on the Struck tracker [31] but more robust from track frame failures

from the use of the Robust Kalman filter to relocate the ROI if the track is lost [8].

- ***Matrioska Best Displacement Flow (MatFlow)*** is a type of motion tracker which is based off of the original Matrioska tracker [44]. It issues the Matrioska tracker in combination with the Best Displacement Flow (BDF) tracker [45]. MatFlow uses the object path given directly by the Matrioska tracker, and since the Matrioska tracker often fails, it is corrected in a low confidence situation with the BDF tracker which tracks the best displacement vector of flow after a clustering of key points is done on the OOI. This tracker is not yet published, but its components have been.

- ***Flock of Trackers (FoT)*** is another type of motion based tracker. FoT estimates the object's trajectory by looking at the estimates given from a number of local trackers that are not robust. These local trackers cover certain specified regions in the frame and then perform their tracking independently. FoT then transforms their displacements and fuses together their estimates to create one [62].

- ***ZHANG Tracker (zhang)*** is a type of feature based tracker. Zhang is a two phase tracker where extracting features and matching them is done. The Zhang tracker uses a dictionary which is built from small overlapping patches from the OOI. This dictionary is then stored and used to compare in the matching phase. The matching phase exposes the same dense patches on each frame and a similarity measure is used to gather a best match in several patches to combine to the ROI. Zhang tracker is not published yet and has only been entered in the VOT challenge to date.

- ***Likelihood of Features Tracking-Lit (loft_lit)*** is another feature based tracker. It extracts image based features and correlation map features and fuses them together in a target appearance model within a search region. It then uses Bayesian maps to estimate the likelihood that each pixel in the search region is part of the OOI. To keep failure rates down, a Kalman filter is used to find the new search region from frame to frame [52, 53, 54]

- ***SumShift (sumshift)*** is a type of probabilistic tracker, furthermore a histogram based tracker. It is an improvement of the mean-shift tracker [14]. It finds histograms on multiple patches of the object model which helps to preserve the geometric structure of the object. It then computes the object likelihood by pooling the probabilities of each path, which increases robustness and accuracy in tracking [41]

- ***Normalized Cross Correlation(NCC)*** is VOT's implemented tracker used to be a baseline to the competition. It uses a simple technique of matching a gray scale template to the new frames in the video sequence using normalized cross-correlation [39].

There were 62 trackers submitted to the VOT 2015 challenge, but we focus on only these eight for our experiments to see how our tracker compares on single video sequences. Then we look where we would place in the entire competition if entered.

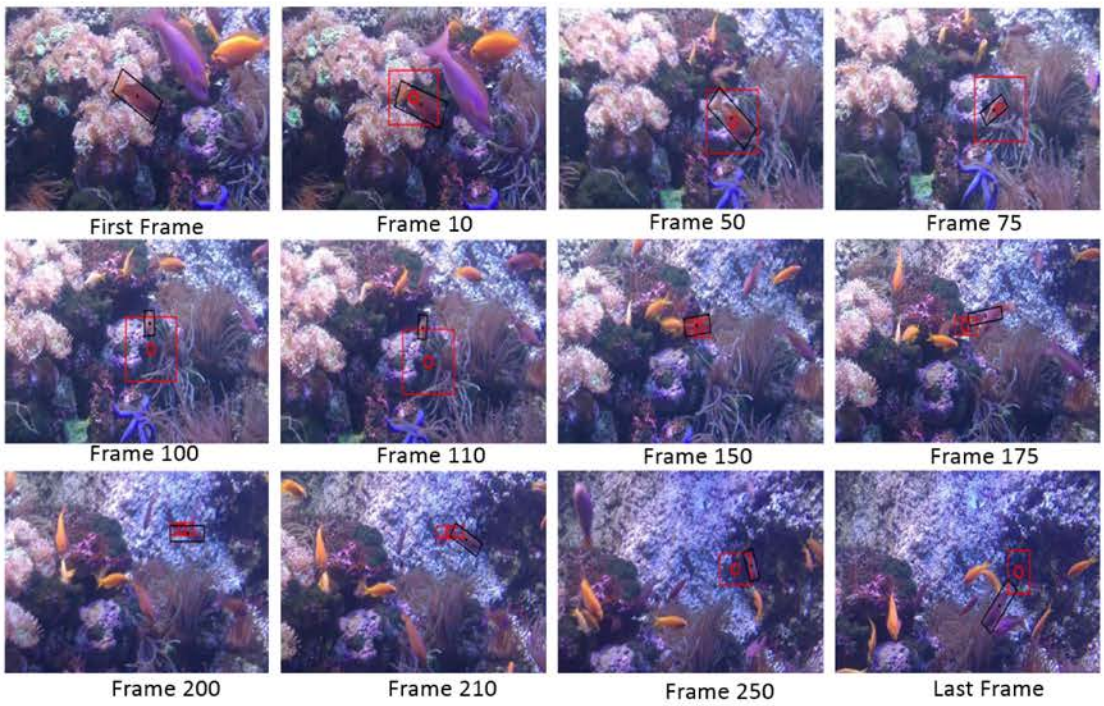

Figure 5.4: Tracking results for VOT's Fish2. Ground truth in black box, tracking ROI in red box

5.4 Results

Our tracker will be compared to the other eight trackers listed in Section (5.3.2) in terms of accuracy overlap and failure rate. Algorithm execution speed is also a measure used in the challenge which will be discussed after tracking performance is addressed.

5.4.1 Individual Video Comparison

Looking at several of VOT's video sequences, we can see some strong results for our tracker. The video shown in the following figure is one of an underwater scene on a coral reef following a particular fish. This video has several obstacles in it, including size change, occlusion, motion change, and camera motion.

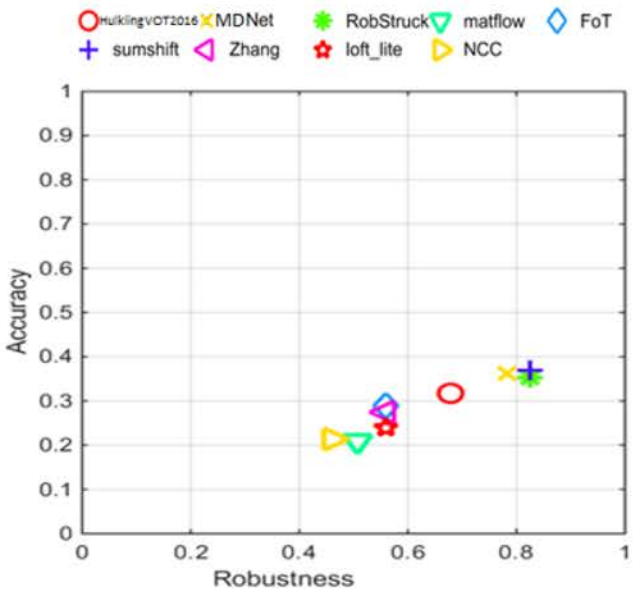

Figure 5.5: Accuracy vs Robustness Plot of Fish2

Looking at Figure (5.4) we can see our tracker stayed with the OOI fairly well. In frame 100-110 it looks like the track is beginning to be lost. Additionally in frame 250 to the last frame (310) it looks like the track is starting to fail. This may be due to the size change in the ROI in terms of scaling and aspect ratio change. Looking at our competition in Figure (5.5) we can see we competed well against the other trackers. The best position in the plot of accuracy versus robustness is the top right corner where accuracy would be 1 and robustness would be 1, meaning you were exactly over the ground truth the entire length of the video. A location in the bottom left would mean you were not on the ground truth at all during the entire video. So being around 32% accuracy compared to the top competitors at 38% is a fairly good track for our proposed tracker. We were fairly robust at only four frame failures during this video.

Since the explanation of the tracking images and plots have been shown, next we will see a series of videos and our tacking ability during the video as well as

Figure 5.6: Visual tracking results on videos from VOT data set

the plots showing our ability against competition. These videos shown below all have multiple obstacles in them. Basketball and Iceskater1 show motion change, occlusion, and size change. They also show a constant obstacle of camera motion throughout the entire video sequence. Birds1 has all of the challenges involved in the video; size change, occlusion, motion change, illumination change, and camera motion. Bolt2 has camera and object motion change throughout the video. Rabbit is quite possibly the hardest video in the data set. The goal is to track a white rabbit in an avalanche situation. It shows constant occlusion, motion chance, and camera motion chance.

Looking at these visual results, our tracker kept up well with the challenges in each of these video sequences. The estimated ROI is similar to size and shape of the ground truth in many of the videos, and shows promising results in competition.

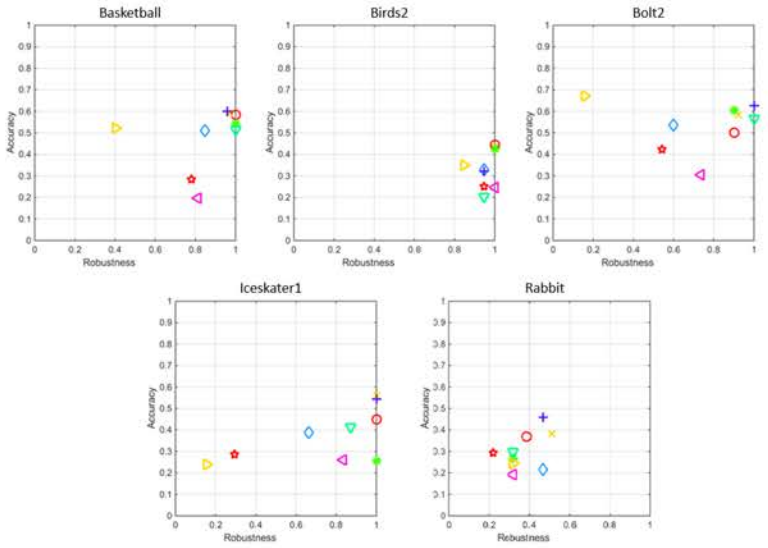

Figure 5.7: Accuracy versus robustness plots for videos shown above

In the videos shown above, our proposed algorithm performs as well as the state of the art trackers and even out performs them in some videos. Our tracker seems to do well tracking humans in situations where object motion, camera motion, and occlusion happens. Our algorithm also performed in the top three trackers with the Rabbit video, which shows constant occlusion throughout the video sequence. Being highly based on color intensities, it is a great result seeing our tracker perform so well on a video where the background is so similar to the foreground OOI. This shows our tracker is robust enough to perform in sequences where the scene is complex and clutter is involved, a highly sought after trait for a tracking algorithm in practice.

Speed was another measure of interest. We wanted our tracker to perform near real time in terms of frames per second. Using GPUs and high performance computing, we achieved that goal

Table 5.2: VOT 2015 Individual video results, seq (sequence), ac(accuracy), f(failures), and sp(speed).

Seq	Ac	F	Sp	Seq	Ac	F	Sp
bag	0.32	1	159.35	**handball1**	0.46	16	118.7
ball1	0.18	9	36.16	**handball2**	0.36	17	88.98
ball2	0.00	5	28.71	**helicopter**	0.20	2	110.69
basketball	0.58	0	219.66	**iceskater1**	0.48	1	201.52
birds1	0.21	20	50.34	**iceskater2**	0.40	1	163.75
birds2	0.32	0	166.34	**leaves**	0.01	7	32.03
blanket	0.41	1	217.85	**marching**	0.29	7	60.67
bmx	0.23	1	61.74	**matrix**	0.25	4	81.41
bolt1	0.41	6	139.59	**motocross1**	0.16	8	80.45
bolt2	0.51	1	226.47	**motocross2**	0.41	1	42.76
book	0.15	12	71.95	**nature**	0.29	2	121.28
butterfly	0.27	3	110.96	**octopus**	0.30	0	119.46
car1	0.27	19	109.39	**pedestrian1**	0.46	7	97.89
car2	0.27	3	214.34	**pedestrian2**	0.35	24	112.49
crossing	0.29	5	66.61	**rabbit**	0.43	5	105.94
dinosaur	0.31	2	171.59	**racing**	0.42	1	151.09
fernando	0.26	4	116.85	**road**	0.35	9	107.48
fish1	0.34	4	167.77	**shaking**	0.49	2	170.74
fish2	0.26	4	163.14	**sheep**	0.31	2	203.99
fish3	0.29	5	183.02	**singer1**	0.30	0	133.29
fish4	0.33	5	214.73	**singer2**	0.46	2	147.56
girl	0.41	6	35.72	**singer3**	0.31	34	13.90
glove	0.20	7	73.26	**soccer1**	0.33	7	148.82
godfather	0.36	3	200.75	**soccer2**	0.21	15	31.89
graduate	0.31	10	37.24	**soldier**	0.29	2	75.22
gymnastics1	0.26	5	36.69	**sphere**	0.20	3	118.87
gymnastics2	0.39	5	68.50	**tiger**	0.33	10	88.17
gymnastics3	0.16	3	49.87	**traffic**	0.30	1	133.03
gymnastics4	0.25	4	125.37	**tunnel**	0.49	1	217.72
hand	0.26	13	106.81	**wiper**	0.41	3	185.48
				Average	0.34	6.04	158.65

As seen in Table (5.2) our tracker excelled past our expectations in terms of speed. The VOT toolkit plays each video as a loop of images, therefore it does not need to be completed in the time it takes to show a video. Meaning, these speed numbers above are the frames per seconds at which our algorithm could perform if a video was not shot with a limited frames per second (fps). For instance, Bag is shot at 20 fps, however, since we do not have to wait for the video to run at 20 fps, we can pull frames out as we are ready to perform calculations, therefore our algorithm performs at 159.3 fps on that particular video. It is important to note, the high definition videos (birds2, gymnastics2, and motorcross2) of the data set were also accomplished with speeds over 20 fps. Our tracker averages 158.65 fps which are speeds beyond real time. This leads us to the assumption that our tracker can perform real time video tracking on ultra high definition video as well as live streaming video. The high speed is due to the GPU implementations made in the algorithm in order to mass calculate per pixel calculations which is the foundation of our tracker.

Now that we have shown our tracker and how well it performs, it is time to show how it would have placed in the VOT 2015 open tracking challenge against 62 other tracking algorithms. The results from VOT 2015 have been shown in Table (**??**).

The trackers in the dataset were ranked based on a new metric, expected overlap accuracy $\hat{\Phi}$ which is a single metric which takes into consideration the accuracy overlap and failures of each tracker. It is important to note, speed is not considered in this expected overlap accuracy. Out of the trackers listed above, MDNet which we compared results to in several videos finished the competition first with $\hat{\Phi} = 0.38$. The baseline tracker, NCC, finished the contest in 62nd place. This is with an

Table 5.3: First half of VOT 2015 Results with proposed tracker (red). M = Matlab, C=C/C++, G=GPU

Rank	Tracker	Accuracy	Failures	Φ	Speed	Framework
1	**MDNet**	0.60	0.69	0.38	0.87	M C G
2	**DeepSRDCF**	0.56	1.05	0.32	0.38	M C
3	**EBT**	0.47	1.02	0.31	1.76	M C
4	**SRDCF**	0.56	1.24	0.29	1.99	M C
5	**LDP**	0.51	1.84	0.28	4.36	M C
6	**sPST**	0.55	1.48	0.28	1.01	M C
7	**SC-EBT**	0.55	1.86	0.25	0.80	M C
8	**NSAMF**	0.53	1.29	0.25	5.47	M
9	**Struck**	0.47	1.61	0.25	2.44	C
10	**RAJSSC**	0.57	1.63	0.24	2.12	M
1	**S3Tracker**	0.52	1.77	0.24	14.27	C
12	**SumShift**	0.52	1.68	0.23	16.78	C
13	**SODLT**	0.56	1.78	0.23	0.83	M C G
14	**DAT**	0.49	2.26	0.22	9.61	M
15	**MEEM**	0.50	1.85	0.22	2.70	M
16	**RobStruck**	0.48	1.47	0.22	1.89	C
17	**OACF**	0.58	1.81	0.22	2.00	M C
18	**MCT**	0.47	1.76	0.22	2.77	C
19	**HMMTxD**	0.53	2.48	0.22	1.57	C
20	**ASMS**	0.51	1.85	0.21	115.09	C
21	**MKCF+**	0.52	1.83	0.21	1.23	M C
22	**TRIC-track**	0.46	2.34	0.21	0.03	M C
23	**AOG**	0.51	1.67	0.21	0.97	*binary*
24	**SME**	0.55	1.98	0.21	4.09	M C
25	**MvCFT**	0.52	1.72	0.21	2.24	*binary*
26	**SRAT**	0.47	2.13	0.20	15.23	M C
27	**Dtracker**	0.50	2.08	0.20	10.43	C
28	**SAMF**	0.53	1.94	0.20	2.25	M
29	**G2T**	0.45	2.13	0.20	0.43	M C
30	**MUSTer**	0.52	2.00	0.19	0.52	M C
31	**TGPR**	0.48	2.31	0.19	0.35	M C
32	**HRP**	0.48	2.39	0.19	1.01	M C

Table 5.4: Second half of VOT 2015 Results with proposed tracker (red). M = Matlab, C=C/C++, G=GPU

Rank	Tracker	Accuracy	Failures	Φ	Speed	Framework
33	**KCFv2**	0.48	1.95	0.19	10.90	M
34	**CMIL**	0.43	2.47	0.19	5.14	C
35	**ACT**	0.46	2.05	0.19	9.84	M
36	**MTSA-KCF**	0.49	2.29	0.18	2.83	M
37	**LGT**	0.42	2.21	0.17	4.12	M C
38	**DSST**	0.54	2.56	0.17	3.29	M C
39	**MIL**	0.42	3.11	0.17	5.99	C
40	**KCF2**	0.48	2.17	0.17	4.60	M
41	**sKCF**	0.48	2.68	0.16	66.22	C
42	**BDF**	0.40	3.11	0.15	200.24	C
43	**KCFDP**	0.49	2.34	0.15	4.80	M
44	**PKLTF**	0.45	2.72	0.15	29.93	C
45	**HoughTrack**	0.42	3.61	0.15	0.87	C
46	**FCT**	0.43	3.34	0.15	83.37	C
47	**MatFlow**	0.42	3.12	0.15	81.34	C
48	**SCBT**	0.43	2.56	0.15	2.68	C
49	**DFT**	0.46	4.32	0.14	3.33	M
50	**FoT**	0.43	4.36	0.14	143.62	C
51	**LT-FLO**	0.44	4.44	0.13	1.83	M C
52	**L1APG**	0.47	4.65	0.13	1.51	M C
53	**OAB**	0.45	4.19	0.13	8.00	C
54	**IVT**	0.44	4.33	0.12	8.38	M
55	**STC**	0.40	3.75	0.12	16.00	M
56	**CMT**	0.40	4.09	0.12	6.72	C
57	**CT**	0.39	4.09	0.11	12.90	M
58	**FragTrack**	0.43	4.85	0.11	2.08	C
59	**ZHANG**	0.33	3.59	0.10	0.21	M
60	**Hulkling**	0.34	6.04	0.08	158.65	C G
61	**LOFT-Lite**	0.34	6.35	0.08	0.75	M
62	**NCC**	0.50	11.34	0.08	154.98	C
63	**PTZ-MOSSE**	0.20	7.27	0.03	18.73	C

expected overlap accuracy of 0.08. Our tracker, based on the accuracy and failures shown in Table (5.2) would give us an expected overlap accuracy $\hat{\Phi}$= 0.08 which puts us in competition with NCC and LOFT_lite. Since our accuracy was the same but our failures were less than LOFT_lite, it is safe to say our tracker would have placed in 60th place out of 63. These results are not expected considering how well the tracker did in the above experiments, however there are some things which hinder our tracker in competition with other trackers which will be discussed later. Our tracker is still one of the fastest performing trackers in the challenge and still one of the more practical trackers to use for real time applications.

5.4.2 Discussion

Using HSV color space works well when looking at humans and trying to track a human object. The hue channel of the HSV color space is the main focus when tracking humans using HSV. This is mostly because the hue channel seems to delineate the human skin color better than just RGB color space. Where HSV histogram tracking breaks down is any other object that isn't human skin. Even with tracking a human, the hue of an image does not vary greatly and a lot of noise is brought into the back projection and the track is lost quickly. It does not do well with occlusion, scaling, lighting, or video resolution quality.

RGB color space gives a more robust back projection with cleaner results than just hue. When looking at the above algorithm, we now have seven back projection probability matrices that are used in creating one to use CAMSHIFT. This brings in more localized information to each pixel, rather than just using hue alone. Also, since hue does a great job in delineating a human face alone, this addition of the red-green, red-blue, and blue-green back projections generates more information

allowing a human face to stand out more. The amount of information gathered by the seven different probability matrices really helps the RGB cooccurrence tracking method when other objects are in question, not necessarily humans. Though RGB tracking seems to perform better on any object and is more robust to occlusion, it does have its downfalls as well. Lighting is a huge issue with the RGB tracking, though the RGB color space has three separate channels with 256 different pixel value intensities to make some linear combination of those three to make one color seems to makes sense to track distinct colors. However, with lighting issues a bright pink shirt on a human or a brown horse running on the beach will all be mixed up by lighting and the algorithm will only see almost white objects and the track will be lost. The same is true with scaling; if the object zooms away from the camera like a red helicopter flying away from an airshow The helicopter's bright red color will become closer to the background due to focal blurring and all apparent robustness it had originally will be gone. Lastly, the resolution of the video plays another role in defeating the RGB method. If our object of interest is blurred due to poor quality video, then we have the same effect as the above scaling and the color distinctiveness is lost to the background or other objects surrounding the object of interest.

Both of the color space probability tracking methods do not show enough characteristics of an object independent of color, like the objects shape or texture. This means the methods described above do not know the difference between an orange cat and an orange, or an orange cat walking the streets and an orange cat in a still picture frame on a wall in a house. This makes tracking even more a challenge to the algorithm.

With the use of our covariance features and the pairwise combinations of RGB

features our tracker was able to overcome some robustness issues faced to the color space histogram tracker outperforms the original tracker. More information about the OOI was gathered and when used in conjunction with the RGB information, our tracker This extra feature in terms of probabilistic histogram tracking really gave our tracker a better chance to fully encapsulate the OOI in the back projection. The added covariance features added a degree of robustness in terms of rotation, camera motion, and occlusion.

Occlusion is where our tracker seems to shine. In many of the videos in the dataset where occlusion is the prominent obstacle, our tracker performed the best compared to the competition. We also fared well with video sequences on humans where motion change in both object and camera were the biggest concern in the video. Our tracker overcoming these obstacles gives promising results for future work in the field.

Where our tracker faulted using our covariance features and RGB color features is when illumination change and object size change was the most prominent obstacles in the video. Our covariance features were not a fusion of multiple features giving information about the objects shape, texture, size, etc so when looking at covariance features of just RGB pixel intensities it is obvious that illumination would be a hindrance. To overcome this, a new color space could have been used, one such as a normalized monochromatic color space or even looking at HSV color spaces in conjunction to our RGB in developing the covariance features. If we were to use features that are scale invariant such as SIFT, SURF, or ORB could help us overcome the object size change issues when tracking.

A point to be made that was eluded to earlier is the fact that our tracker is model free. There is no need to pre-train our tracker on learned features or char-

acteristics of the OOI before we begin a tracking. The case of the top contenders in the VOT challenge were not model free and had ground truth data of the actual video sequences to pre-train their tracker. This pre-training could come into use if you knew, for example, that you wanted to track a red car and only a red car during a high speed chase for use of law enforcement. But to say that law enforcement only wants to track a red car is unrealistic. What if the culprit escapes the car and begins fleeing on foot. Our tracker would be able to track the car during the chase and then the human as soon as they flee the vehicle and be able to do it all real time. When looking at this, it is easy to say this is an unfair advantage to the competition. However, since our tracker did outperform some of these top trackers in a few video sequences helps solidify our proposed algorithm as a strong probabilistic tracker in the community. Also to note on these top performing trackers, the speed in which they execute is impractical for any real time use. As a matter of fact, most of the trackers performed in times slower than real time, and some slower than a frame per second. This can't be used in any real life applications where tracking real time is needed. These trackers are great for a static competition, but that is as far as they go in terms of usefulness. Our tracker however could be used for real time surveillance and tracking and has shown to be competitive with new state of the art trackers.

Chapter 6

Conclusion

In this thesis we presented an integrated approach for probabilistic tracking that combines appearance, shape models, topology constraints, and efficient sampling. The integrated approach will extracts features per pixel using a neighborhood scheme. The concatenation of the dynamic feature pool into a single feature descriptor is the strength of our approach. The novel approach fits into an established tracking algorithm which we generalized to D dimensions in order to create a more robust tracker.

This algorithm was implemented on a GPU utilizing high performance computing to achieve real time video tracking. We tested our algorithm on a benchmark data set to illustrates that our proposed solution is competitive when compared to other state-of-the-art trackers. Even though our proposed algorithm did not outperform the winning algorithms in the benchmark competition in expected overlap accuracy, it did exceed the state-of-the-art trackers in speed.

In future work there are several gaps that need to be addressed. We will investigate additional features to improve the accuracy and failure rate results. Another

improvement would to add more consistent method for keeping a track on the object. This could entail a closed contour, or rotating bounding box. The continued effort is to improve the speed performance. Regardless of the algorithm complexity, maintain real time capability. Therefore, deploying GPU implementations for all feature extractions is important and under continuous development.

Bibliography

[1] Amit Adam, Ehud Rivlin, and Ilan Shimshoni. Robust fragments-based tracking using the integral histogram. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 798–805. IEEE, 2006.

[2] Vincent Arvis, Christophe Debain, Michel Berducat, and Albert Benassi. Generalization of the cooccurrence matrix for colour images: application to colour texture classification. *Image Analysis & Stereology*, 23(1):63–72, 2011.

[3] Garvit Arya, Manisha Singh, and Mayank Gupta. Human-computer interaction based on real-time motion gesture recognition. *Human-Computer Interaction*, 4(3), 2016.

[4] Shai Avidan. Support vector tracking. *IEEE transactions on pattern analysis and machine intelligence*, 26(8):1064–1072, 2004.

[5] Florian Bartels. Vehicle navigation system, February 11 2016. US Patent 20,160,040,994.

[6] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008(1):1–10, 2008.

[7] Horn BKP. Robot vision, 1986.

[8] Ivan Bogun. *Robust Structured Tracking*. PhD thesis, Florida Institute of Technology, 2016.

[9] Gary R Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, 1998.

[10] Ted J Broida, S Chandrashekhar, and Rama Chellappa. Recursive 3-d motion estimation from a monocular image sequence. *IEEE Transactions on Aerospace and Electronic Systems*, 26(4):639–656, 1990.

[11] Kevin Cannons and Richard Wildes. Spatiotemporal oriented energy features for visual tracking. In *Asian Conference on Computer Vision*, pages 532–543. Springer, 2007.

[12] Peng Chang and John Krumm. Object recognition with color cooccurrence histograms. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2. IEEE, 1999.

[13] Meng Chee, Paul Dixon, Florian Dusch, Hyung Keun Kim, Luyza Viana Pereira, Bartley H Calder, Scott Barrow, Sudhir Kumar Misra, and Gabriel Nicolae. Video editing using contextual data and content discovery using clusters, April 14 2016. US Patent 20,160,104,508.

[14] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.

[15] Changhyun Choi and Henrik I Christensen. Real-time 3d model-based tracking using edge and keypoint features for robotic manipulation. In *Robotics and*

Automation (ICRA), 2010 IEEE International Conference on, pages 4048–4055. IEEE, 2010.

[16] Jin Woo Choi, Taeg Keun Whangbo, and Cheong Ghil Kim. A contour tracking method of large motion object using optical flow and active contour model. *Multimedia Tools and Applications*, 74(1):199–210, 2015.

[17] Patrick Pakyan Choi and Martial Hebert. Learning and predicting moving object trajectory: a piecewise trajectory segment approach. *Robotics Institute*, page 337, 2006.

[18] David A Clausi. An analysis of co-occurrence texture statistics as a function of grey level quantization. *Canadian Journal of remote sensing*, 28(1):45–62, 2002.

[19] Robert T Collins. Mean-shift blob tracking through scale space. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–234. IEEE, 2003.

[20] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[21] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 142–149. IEEE, 2000.

[22] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[23] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Learning spatially regularized correlation filters for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4310–4318, 2015.

[24] Jonathan Deutscher, Andrew Blake, and Ian Reid. Articulated body motion capture by annealed particle filtering. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 126–133. IEEE, 2000.

[25] Ernst Dieter Dickmanns and Volker Graefe. Dynamic monocular machine vision. *Machine vision and applications*, 1(4):223–240, 1988.

[26] Ruofei Du, Sujal Bista, and Amitabh Varshney. Video fields: fusing multiple surveillance videos into a dynamic virtual environment. In *Proceedings of the 21st International Conference on Web3D Technology*, pages 165–172. ACM, 2016.

[27] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Computer science and scientific computing. Academic Press, Boston, 1990.

[28] Ryan A Gill, W Andrew Cox, and Frank R Thompson III. Timing of songbird nest predation as revealed by video surveillance. *The Wilson Journal of Ornithology*, 128(1):200–203, 2016.

[29] Robert M Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein. Textural features for image classification. *Systems, Man and Cybernetics, IEEE Transactions on*, (6):610–621, 1973.

[30] Robert M Haralick, Karthikeyan Shanmugam, et al. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, 3(6):610–621, 1973.

[31] Sam Hare, Amir Saffari, and Philip HS Torr. Struck: Structured output tracking with kernels. In *2011 International Conference on Computer Vision*, pages 263–270. IEEE, 2011.

[32] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[33] Michael Isard and Andrew Blake. Condensation conditional density propagation for visual tracking. *International journal of computer vision*, 29(1):5–28, 1998.

[34] Wallace Jackson. The automation of digital video: Programming. In *Digital Video Editing Fundamentals*, pages 153–166. Springer, 2016.

[35] Yong-Hyun Jang, Jung-Keun Suh, Ku-Jin Kim, and Yoo-Joo Choi. Robust target model update for mean-shift tracking with background weighted histogram. *KSII Transactions on Internet & Information Systems*, 10(3), 2016.

[36] Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In *AeroSense'97*, pages 182–193. International Society for Optics and Photonics, 1997.

[37] David R Karger. Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In *SODA*, volume 93, pages 21–30, 1993.

[38] Zia Khan, Tucker Balch, and Frank Dellaert. A rao-blackwellized particle filter for eigentracking. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–980. IEEE, 2004.

[39] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Luka Cehovin, Gustavo Fernandez, Tomas Vojir, Gustav Hager, Georg Nebehay, and Roman Pflugfelder. The visual object tracking vot2015 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1–23, 2015.

[40] Matej Kristan, Janez Perš, Matej Perše, and Stanislav Kovačič. A bayes-spectral-entropy-based measure of camera focus using a discrete cosine transform. *Pattern Recognition Letters*, 27(13):1431–1439, 2006.

[41] Jae-Yeong Lee and Wonpil Yu. Visual tracking by partition-based histogram backprojection and maximum support criteria. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 2860–2865. IEEE, 2011.

[42] Sukwon Lee. Human computer interaction using eye-tracking data. *Human Computer Interaction*, 2016.

[43] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[44] Mario Edoardo Maresca and Alfredo Petrosino. Matrioska: A multi-level approach to fast tracking by learning. In *International Conference on Image Analysis and Processing*, pages 419–428. Springer, 2013.

[45] Mario Edoardo Maresca and Alfredo Petrosino. Clustering local motion estimates for robust and efficient object tracking. In *European Conference on Computer Vision*, pages 244–253. Springer, 2014.

[46] Peter S. Maybeck. *Stochastic models, estimation, and control*, volume 141 of *Mathematics in Science and Engineering*. 1979.

[47] Mark Moyou, Koffi Eddy Ihou, Rana Haber, Anthony Smith, Adrian M Peter, Kevin Fox, and Ronda Henning. Bayesian fusion of back projected probabilities (bfbp): Co-occurrence descriptors for tracking in complex environments. In *Advanced Concepts for Intelligent Vision Systems*, pages 167–180. Springer, 2015.

[48] Mark Moyou, Koffi Eddy Ihou, Rana Haber, Anthony Smith, Adrian M Peter, Kevin Fox, and Ronda Henning. Bayesian fusion of back projected probabilities (bfbp): Co-occurrence descriptors for tracking in complex environments. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 167–180. Springer, 2015.

[49] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. *arXiv preprint arXiv:1510.07945*, 2015.

[50] CUDA Nvidia. Programming guide, 2008.

[51] PR Ouyang, Truong Dam, J Huang, and WJ Zhang. Contour tracking control in position domain. *Mechatronics*, 22(7):934–944, 2012.

[52] Kannappan Palaniappan, Filiz Bunyak, Praveen Kumar, Ilker Ersoy, Stefan Jaeger, Koyeli Ganguli, Anoop Haridas, Joshua Fraser, Raghuveer M Rao, and Guna Seetharaman. Efficient feature extraction and likelihood fusion for vehicle tracking in low frame rate airborne video. In *Information fusion (FUSION), 2010 13th Conference on*, pages 1–8. IEEE, 2010.

[53] Rengarajan Pelapur, Sema Candemir, Filiz Bunyak, Mahdieh Poostchi, Guna Seetharaman, and Kannappan Palaniappan. Persistent target tracking using likelihood fusion in wide-area and full motion video sequences. In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 2420–2427. IEEE, 2012.

[54] Rengarajan Pelapur, Kannappan Palaniappan, and Gunasekaran Seetharaman. Robust orientation and appearance adaptation for wide-area large format video object tracking. In *Advanced Video and Signal-Based Surveillance (AVSS), 2012 IEEE Ninth International Conference on*, pages 337–342. IEEE, 2012.

[55] Fatih Porikli, Oncel Tuzel, and Peter Meer. Covariance tracking using model update based on lie algebra. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 728–735. IEEE, 2006.

[56] via Wikimedia Commons Ricardo Cancho Niemietz [Public domain]. Rgb 24bits palette sample image, 2008. https://commons.wikimedia.org/wiki/File

[57] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, volume 23, pages 309–314. ACM, 2004.

[58] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. IEEE, 2011.

[59] David Sharp, Craig Stoneking, and Kingsley Fregene. Micro air vehicle based navigation aiding in degraded environments. In *2016 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 305–312. IEEE, 2016.

[60] Oncel Tuzel, Fatih Porikli, and Peter Meer. Region covariance: A fast descriptor for detection and classification. In *European conference on computer vision*, pages 589–600. Springer, 2006.

[61] Luc Van Gool, Piet Dewaele, and André Oosterlinck. Texture analysis anno 1983. *Computer vision, graphics, and image processing*, 29(3):336–357, 1985.

[62] Tomáš Vojíř and Jiří Matas. The enhanced flock of trackers. In *Registration and Recognition in Images and Videos*, pages 113–136. Springer, 2014.

[63] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158. Ieee, 2000.

[64] Junqiu Wang and Yasushi Yagi. Integrating color and shape-texture features for adaptive real-time object tracking. *IEEE Transactions on Image Processing*, 17(2):235–240, 2008.

[65] Changjiang Yang, Ramani Duraiswami, and Larry Davis. Efficient mean-shift tracking via a new similarity measure. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 176–183. IEEE, 2005.

[66] Ju Hong Yoon, Ming-Hsuan Yang, Jongwoo Lim, and Kuk-Jin Yoon. Bayesian multi-object tracking using motion context from multiple objects. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 33–40. IEEE, 2015.

[67] Huiyu Zhou, Yuan Yuan, and Chunmei Shi. Object tracking using sift features and mean shift. *Computer vision and image understanding*, 113(3):345–352, 2009.