APPEARANCE MODELING FOR PERSISTENT OBJECT TRACKING IN WIDE-AREA & FULL MOTION VIDEO

A Thesis presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

---

by

RENGARAJAN PELAPUR

Dr. K. Palaniappan, Thesis Supervisor

July 2016

The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation entitled:

APPEARANCE MODELING FOR PERSISTENT OBJECT TRACKING
IN WIDE-AREA & FULL MOTION VIDEO

presented by Rengarajan Pelapur,
a candidate for the degree of Doctor of Philosophy and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Kannappan Palaniappan

Dr. Jeffrey Uhlmann

Dr. Filiz Bunyak Ersoy

Dr. Xu Han

ACKNOWLEDGMENTS

This thesis is the culmination of five years of research in appearance based persistent target tracking that began with a proposal for a unique fusion based tracking method that was in turn modified and re-implemented into a flexible and easy to modify framework. My basics of Computer Vision and Image processing were based on five full time graduate level courses that were taught by Prof. Tony Han and Prof. Ye Duan. The techniques that I used in improving the LOFT algorithm primarily came from ideas that were a direct consequence of understanding the basics and wanting to do beyond what is given as assignments during the coursework. For this and more, I would like to thank Prof. Tony Han and Prof Ye Duan. Dr. Filiz Bunyak provided a full overview and showed extreme patience in the initial days of when I started working on LOFT. I started using interpreted languages to prototype my research ideas and showcase preliminary results in order to balance the improvement of results versus the amount of time that I would have spent otherwise in fruitless endeavors. I associate this pragmatic approach to Dr. Bunyak as she would spend innumerable hours and help explain the outcome by pouring over the rough results that were generated. I would like to thank Dr. Bunyak for teaching me essential research skills, which while I am still learning and improving, has been instrumental in putting the dissertation work together. I learned a lot of dynamics and Kalman filtering from the very best in the field such as Prof. Jeffrey Uhlmann and Prof. Simon Julier. The entire Kalman filtering code was written during regular coursework that Dr. Uhlmann taught at graduate level. His feedback during my Comprehen-

sive exam has helped shape my dissertation. I am grateful to Prof. Uhlmann and Prof. Julier for giving me the opportunity to interact and work with them. None of this would have been possible without the invaluable feedback from my advisor Prof. Kannappan Palaniappan. Prof. Palaniappan has followed my progress very closely and has always been actively involved, be it the theoretical work or in engineering and actual coding, giving me feedback at every step. His meticulous editing of my written work has helped me publish and present in a detailed yet simplistic manner that has largely resulted in positive reviews from the academic community. I would also like to thank my very good friend Dr. Brittany Morago who has been in the same computer vision and machine learning classes as I have since I joined the university. The competition to always do better in assignments helped me focus on not just the quality of results but also on better engineering my code. I learned a lot about the academic world from Dr. Surya Prasath and started focusing more on mathematical image processing which peaked my interest in the final few years of my doctoral degree. His contribution to my understanding of sparse systems, anisotropic filtering, global minimization algorithms and so much more cannot be easily quantified. Raphael Viguier, due to his strong engineering basics, has always helped me push myself to read and strengthen my own basics. Dr. Ilker Ersoy helped me with the code integration with Kitware and I always enjoy the intellectual discussions. Anoop Haridas, with his work on Kolam, has assisted in a lot of the visualizations in this dissertation work. His regular detailed questions and feedback about data structures and programming has pushed me to read more about C++ and become a better programmer in a language that has always been my favorite. I am also thankful to

the CIVA lab members such as Mahdieh Pootschi, Ke Gao, Shizeng Yao for helping me test the LOFT code. I learned a lot of the 3D reconstruction techniques with hands on practical coding from Dr. Hadi Aliakbarpour. The collaboration with him has taught me more about a different field within computer vision. I would like to thank Urmila Sampathkumar for helping me through the last few months leading up to my defense and also for the encouraging words as I interviewed for potential positions. Her patience, while I raved on about the dissertation content during the preparation of my defense and to this day, is something that I take for granted. Our semi-weekly CIVA lab meetings discussed a wide variety of topics in image processing, computer vision and machine learning. While to newcomers the meetings would be downright intimidating, I learned a lot in putting my point across effectively and generally presenting in an inclusive manner. The lab's diversity, academic backgrounds and in other ways, has taught me the importance of respecting everyone's opinions and using the criticism in a constructive way to improve my own work.

Dr. Arslan Basharat has always provided valuable feedback on the LOFT system and integration during the CETE project and was my manager during my summer internship at Kitware. I would like to thank Dr. Guna Seetharaman for his continued support for the presence of CIVA lab's algorithms like LOFT in the CETE project. His recommendations have helped me with experiments of different techniques, particularly a specific implementation of the Fourier transform for orientation estimation. Lastly I would like to thank the countless friends and CIVA lab members who have indirectly or directly contributed to this work. A special thanks to my parents, my uncle, my brother and sister in law for supporting my graduate school ambitions and

for their infinite patience and help throughout my academic career to this day. This research was partially supported by U.S. Air Force Research Laboratory (AFRL) under agreement AFRL FA8750-11-C-0091, FA8750-14-2-0072 and FA8750-14-C-0044 and contains work that was approved for public release (cases 88ABW-2012-1013 & 88ABW-2012-3914). The views and conclusions contained in this dissertation work are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of AFRL or the U.S. Government.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

ABSTRACT

Object tracking is a core element of computer vision and autonomous systems. As such single and multiple object tracking has been widely investigated especially for full motion video sequences. The acquisition of wide-area motion imagery (WAMI) from moving airborne platforms is a much more recent sensor innovation that has an array of defense and civilian applications with numerous opportunities for providing a unique combination of dense spatial and temporal coverage unmatched by other sensor systems. Airborne WAMI presents a host of challenges for object tracking including large data volume, multi-camera arrays, image stabilization, low resolution targets, target appearance variability and high background clutter especially in urban environments. Time varying low framerate large imagery poses a range of difficulties in terms of reliable long term multitarget tracking. The focus of this thesis is on the Likelihood of Features Tracking (LOFT) testbed system that is an appearance based (single instance) object tracker designed specifically for WAMI and follows the track before detect paradigm. The motivation for tracking using dynamics before detecting was so that large scale data can be handled in an environment where computational cost can be kept at a bare minimum. Searching for an object everywhere on a large frame is not practical as there are many similar objects, clutter, high rise structures in case of urban scenes and comes with the additional burden of greatly increased computational cost. LOFT bypasses this difficulty by using filtering and dynamics to constrain the search area to a more realistic region within the large frame and uses multiple features to discern objects of interest. The objects of interest are expected

as input in the form of bounding boxes to the algorithm. The main goal of this work is to present an appearance update modeling strategy that fits LOFT's track before detect paradigm and to showcase the accuracy of the overall system as compared with other state of the art tracking algorithms and also with and without the presence of this strategy. The update strategy using various information cues from the Radon Transform was designed with certain performance parameters in mind such as minimal increase in computational cost and a considerable increase in precision and recall rates of the overall system. This has been demonstrated with supporting performance numbers using standard evaluation techniques as in literature. The extensions of LOFT WAMI tracker to include a more detailed appearance model with an update strategy that is well suited for persistent target tracking is novel in the opinion of the author. Key engineering contributions have been made with the help of this work wherein the core LOFT has been evaluated as part several government research and development programs including the Air Force Research Lab's Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance (C4ISR) Enterprise to the Edge (CETE), Army Research Lab's Advanced Video Activity Analytics (AVAA) and a proposed fine grained distributed computing architecture on the cloud for processing at the edge. A simplified version of LOFT was developed for tracking objects in standard videos and entered in the Visual Object Tracking (VOT) Challenge competition that is held in conjunction with the leading computer vision conferences. LOFT incorporating the proposed appearance adaptation module produces significantly better tracking results in aerial WAMI of urban scenes.

Chapter 1

Introduction

Motion imagery has become a vital part of intelligence, surveillance and reconnaissance (ISR) in recent years. Data collection capabilities have increased from being rare a decade ago to a regular deployment stage in theater. Increasingly city wide data collects are used for large scale surveillance during event gatherings where maintaining safety and security is difficult. Wide-Area Motion Imagery (WAMI) provides a hawk-eye view of an entire area making it extremely valuable when augmented with standard closed-circuit television (CCTV) surveillance. Persistent large scale object tracking has been a challenging problem. Though several commercial and academic solutions exist, these are predominantly not focused on persistent tracking. Tracking all moving objects or objects in a given area of interest (AOI) are typically the scenarios best suited for such methods. Given the high performance requirement in such cases, a motion based, dynamics oriented approach is taken wherein one would identify all moving objects and then later solve the problem of data association. While

these techniques do quite well and have proven their robustness in terms of lessening the number of track switches (distractors) the appearance component has largely been sidelined due to the difficulty in effective appearance modeling.

1.1 Goals

Likelihood of Features Tracking (LOFT) system as a tracker has been used to track objects in general everyday video with high frame rate and also on really low frame rate video such as WAMI. While we specifically handle challenges that pertain to vehicle tracking in wide-area video, LOFT has also proven to be quite robust in general as well as Full Motion video (FMV) There are two goals to this work. First is to show that appearance based tracking in WAMI can result in promising results. Second is demonstrating how this can be achieved with techniques that address common issues that are part and parcel of explicit appearance modeling techniques. We have supplemented all findings by using state of the art metrics for reporting the results that are part of literature and also have gone beyond the standard accepted metrics for a more through and detailed analysis.

1.2 Challenges

1.2.1 Wide-Area Motion Imagery

- Low and variable frame rate imagery

  Wide-Area Motion Imagery (WAMI) is characterized by extremely low frame

rate of anywhere from $1 - 4$ hertz. Due to hardware constraints that relate to on-board data storage and the high pixel resolution of the camera systems sometimes frame rate is sacrificed in order to get critical data. This results in dropped frames or a consistently variable frame rate that impacts prediction models on a scale and presents a challenge of its own.

- Occlusions

  Typically, occlusions are characterized into two broad categories such as 1) Partial and 2) Full occlusion. Partial occlusions occur due to a wide variety of reasons such as

  - Shadows due to foliage or tall structures such as buildings in urban environments (severe illumination changes)
  - Foliage such as trees with sparse branches that show a completely noisy appearance of the object capable of throwing off the best of the appearance models
  - Bright structures on the road such as the paint on the crosswalks (rapid illumination change)

  A full occlusion occurs when the object that is being tracked is hidden completely from view due to a really tall structure such as a building or an advertisement banner.

- Distractors

  While wide-area imagery is characterized by a large amount of pixel data, the resolution, or commonly referred to as ground sampling distance (GSD) is on

the order of $40 - 65$ cm. The typical orbiting flight height is around 1800 m or 6000 feet. The support map for an object on the ground such as a vehicle is thus reduced to around 20 pixels wide and about a maximum of 40 pixels in height. This coupled with distortions that are caused by camera geometry and the orbiting aerial platform, objects at such a small scale appear very similar to each other.

- Very large amount of data

  Datasets range from having $10 - 20$ million pixels to gigapixel imagery for some multi-camera systems. Appearance matching cannot be performed on such a large scale and thus dynamics, prediction and filtering are essential for designing a real time and more practical system.

1.2.2 FMV and Standard Video

- Appearance change due to affine transformations

  Tracking in standard hand-held video is characterized by appearance changes due to 3D motion of the object. FMV and standard video is usually $24 - 35$ Hz and while the appearance changes are more gradual they are also complex such as a face that goes through all the various poses possible.

- Occlusions

  Occlusions, common with WAMI, also is a challenge in FMV and standard video. Rapid appearance changes along with occlusions can be very difficult to handle without explicit appearance modeling.

- Dynamic background

  Dynamic background such as snow, rain and other weather effects affect FMV and standard video. This can make it very difficult for trackers which only model the foreground object to not become confused with rapidly changing background.

In this dissertation, we have tried to address most of these challenges by proposing a tracking pipeline that is tailored to handle specific difficulties such as orientation change by using a novel usage of the Radon transform, low pixel level resolution by using multiscale features, occlusions by switching to prediction when appearance information is not reliable and engineering a platform that is flexible enough such that we have a more standardized way for researchers to leverage and study the impact of individual modules on overall tracking quality on different types of datasets. The rest of the thesis is organized as follows. Chapter 2 briefly covers the literature in tracking. WAMI tracking literature is very different than tracking in standard imagery. The clustering of techniques on one side or the other can clearly be seen in this literature survey. Chapter 3 proposes our novel tracking pipeline. LOFT as a feature fusion and appearance modeling system is highly modular. As a first step, the target and search region need to be described by a robust feature set. This is followed by a matching step, where in our case is comprised of several one-to-one feature matches followed by a fusion process that aggregates the matching information to produce a single probability map. Other modules that help with automatic termination and intelligent handling of special cases such as missing or corrupt data is also described. Chapter 4 introduces the radon transform along with the equations for a particular

case. This describes and motivates the use of this technique for our orientation estimation. Chapter 5 describes our proposed LOFT pipeline with the addition of the appearance modeling that is derived from orientation estimation. It reviews the techniques and usage of the radon transform in detail followed by algorithms that describe how this information is converted into descriptors that help with the appearance modeling. Chapter 6 introduces our engineering implementation of the algorithm along with its modules. The integration work with other systems under different programs are also described in detail. The modular architecture is shown along with an interaction diagram. A proposed new method to plug in extra 3D information is also shown that could potentially lead to better tracking results in aerial imagery. Chapter 7 shows our experimental methodology and introduces the data along with the final results. The interpretation of the performance numbers is detailed and we have shown how LOFT outperforms various trackers in literature along with a detailed performance table on the amount of improvement with the added appearance modeling technique. Chapter 8 shows our collaborative effort in running LOFT on a cloud computing architecture. It primarily focuses on evaluation of LOFT in a streaming imagery pipeline for disaster scenarios while running in networked mode. Resource allocation using the cloud computing environment and running a data input heavy algorithm like LOFT would be a step towards making tracking more pervasive. This chapter focuses on the engineering aspects of such a system along with experiments that show the viability of scalable tracking. The final chapter concludes the proposed work and discusses future directions.

Chapter 2

Related Work

Tracking in Wide-Area Motion Imagery (WAMI) poses a number of additional difficulties for vision-based tracking algorithms due to very large gigapixel sized images, low frame rate sampling, low resolution targets, limited target contrast, foreground distractors, background clutter, shadows, static and dynamic parallax occlusions, platform motion, registration, mosacing across multiple cameras, object dynamics, etc. [1–11]. These difficulties make the tracking task in WAMI more challenging compared to standard ground-based or even narrow field-of-view (aerial) full motion video (FMV). Full motion video and standard video has its own set of difficulties where appearance modeling, typically machine learning based techniques do a lot better that template based matching methods. In the next few sections in this chapter we will cover all the varied set of trackers including those specifically designed for WAMI as well as the most recent robust trackers for FMV and standard video.

2.1 Overview

Traditional visual trackers either use motion/change detection or template matching. Motion and appearance based techniques dominate the literature and are used in a variety of videos. Machine learning methods in appearance based techniques are more popular currently for tracking in complex videos. Persistent tracking using motion detection-based schemes need to accommodate dynamic behaviors where initially moving objects can become stationary for short or extended time periods, then start to move again. Motion-based methods face difficulties with registration, scenes with dense set of objects or near-stationary targets. Accuracy of background subtraction and track association dictate the success of these tracking methods [7, 8, 12, 13]. Template trackers on the other hand, can drift off target and attach themselves to objects that seem similar, without an update to the appearance model [14, 15].

Visual tracking is an active research area with a recent focus on appearance adaptation, learning and sparse representation. Appearance models are used in [16–19], classification and learning techniques have been studied in [20, 21], and parts-based deformable templates in [22]. Gu *et al.* [19] stress low computation cost in addition to robustness and propose a simple yet powerful Nearest Neighbor (NN) method for real-time tracking. Online multiple instance learning (MILTrack) is used to achieve robustness to image distortions and occlusions [20]. The P-N tracker [21] uses bootstrapping binary classifiers and showcases reliability by generating lengthier tracks. Mei *et al.* [23, 24] propose a robust tracking method using a sparse representation approach within a particle filter framework to account for pose changes. While there is a lot of variety in the appearance based tracking literature, they are centered around

a few key techniques that are discussed in the next section. Wide-Area and aerial imagery trackers focus more on dynamics and detection. There are a few algorithms that add an additional appearance component while some have a clever technique to solve for the data association problem which is described in the following subsections.

2.2 Model-free Tracking and Appearance models

Model-free tracking is a paradigm where an object of interest is manually marked and the algorithm is expected to track the object reliably through the variety of appearance changes[25]. Model-free tracking remains to be a challenging problem and due to the limited information in the input, several algorithms have devised methods to leverage this in multiple ways where MIL-track [20] samples around the object for foreground and background patches and L1 [26] creates an array of templates which are a result of a warping function that can accommodate the appearance changes. Adaptive appearance models are rare in complex imagery as there is always the unsolved problem of drift. An appearance based tracker is said to have 'drifted' if it eventually adapts to the background due to a defective match. Several methods have been proposed to alleviate this error [27]. A number of tracking strategies such as Robust Fragments-based Tracking (FragTrack) [28], Multiple Instance Learning (MILTrack)[20] exist which can track reliably in the face of complex scenarios such as occlusions, distractors and objects which have not very many distinctive features. These methods have worked well for hand-held camera videos at 60 frames per second. Smuelders et al. [29] recently ran a comprehensive evaluation over the

ALOV++(Amsterdam Library of Ordinary Videos for tracking) dataset. This dataset comprises of real-life videos from YouTube with several different types of targets. The aim of this survey was to publish results of cutting edge methods in tracking literature on real-life data that does not focus on a particular type or kind of scenario. Two highest performing trackers on this dataset were TLD [30] (Tracking, Learning Detection) and Struck [31] (Structured output tracking with kernels).

2.3 Tracking in Wide-Area and Aerial Imagery

For Wide-Area class of aerial data trackers that do not explicitly handle the large amount of data with low pixel level resolution do not always perform well. Motion-Based trackers such as Reilly et al. [8], Saleemi et al. [32] achieved good results on similar WAMI data. Multi-target tracking has a rich body of literature from the computer vision community. The community has only more recently begun to focus on tracking in WAMI from aerial platforms [10, 33–36]. In Prokaj's work [36] the goal was to detect shorter tracks and then associate them to existing tracks. The shorter tracks are determined using background subtraction after registration of aerial imaging. Pollard's [35] work is similar with the exception that an additional false alarm suppression step is performed by using learning of edges of background. Keck [34] focused on a real-time engineering tracking system that processes tiles separately with three frame differencing for the motion detection using efficient box filters. Xiao et al. [37] had a different approach wherein they used the road network

for modeling the dynamical behavior of vehicles and also a detection framework for slow moving or stopped objects. The detections are associated using the classic Hungarian algorithm. There are spatial constraints in place where the velocity and distance difference is preserved between vehicles on subsequent frames. However, this may cause wrong associations as applying such constraints uniformly across all objects due to complex vehicle dynamics. Appearance modeling is using a simple template approach that may not be as comprehensive in terms of detecting a varied set of features to be effective under different imaging conditions.

Reilly et al. [8] propose the use of the Hungarian algorithm for detection association and computing such associations only in smaller image cells in order to be more efficient. Using spatial context, velocity orientation, orientation of the road between detected objects a matching cost is estimated. It is not clear how more complex vehicle dynamics such as a stop or yield situations would cause correct associations. Tracking in aerial imagery which in turn inspired WAMI trackers is a similar problem with the main difference being the difference in resolution and also the lesser number of pixels to be pushed through a detection and tracking pipeline. Yu et al. [38] used a general motion pattern in 4D space using position and velocity in 2D (x,y,vx,vy). This voting framework is used to detect and segment motion patterns in this space. A mean-shift algorithm is then used to stitch the tracks after the initial blob segmentation. In this method a much longer sequence is needed to detect such motion patterns in the 4D space. This initial work may not be quite as realtime due to this requirement. Huang et al. [39] had an approach which would simultaneously detect and track the objects of interest. In a feedback loop, the tracking results were then used

as input to a detection module that would adjust thresholds in motion segmentation to improve the detection performance. This method is sensitive to initial conditions as a bad initialization could be irreversible resulting in poor detection performance. Li et al. [40] used the road network which was extracted first which in turn are the regions of interest. A registration method is proposed where the lane markers are used for detection as well as a mean-shift based tracking method. Basharat et al. [41] also use a Hungarian algorithm for association from frame to frame but also add an appearance based hybrid approach for stitching slow moving or stopped objects. This appearance based technique in future tests used our LOFT algorithm for better results. Ling et al. [9] did a comprehensive evaluation of multiple popular visual trackers on CLIF data followed by our work [42] which showed improvement in the scores over other methods.

2.4 Trackers for Standard Video

There is an abundance of trackers for standard video in literature due to the growth in popularity of the Visual Object Tracking Challenge [43]. The overwhelming response of the tracking community has produced outstanding results on challenging every-day video. These trackers are generally classified as short term trackers. The VOT Challenge has contributed to a more comprehensive evaluation platform as previous evaluations such as the Online Object Tracking (OTB) [44] and ALOV [29] would initialize the algorithms on the first frame and letting the trackers run till the end of the every sequence. Trackers are restarted on a significant drift when compared

against groundtruth. This challenge has featured a very wide gamut of tracking algorithms from traditional correlation based to convolutional neural networks with descriptor and learning based approaches. The rest of this section focuses on introducing standard as well as cutting edge techniques designed and run on standard video.

Normalized cross correlation (NCC), that can be attributed to Lucas et al. [45], uses the appearance of objects from previous frames to match in the current frame simply by correlating a template with a search region. It is also usually the baseline as it is the simplest among all techniques. The Kalman Appearance Tracker [46] uses prediction as its main approach towards handling appearance change. A normalized template is represented by its intensities in a Kalman filter and it predicts the intensity change over time. Target motion is independently predicted by a different filter that maintains the motion dynamics. Candidate windows around the predicted target position are then compared to the predicted template appearance and as per the greedy approach a best match is picked as the most likely location. Fragments-based Robust Tracking (Frag-Track) [47] pursues matching an ensemble of patches. Partial occlusions and pose changes are handled patch-by-patch. Candidate windows are selected uniformly around the previous position similar to the NCC tracker. Incremental Visual Tracking (IVT) [48] is a method in which stores the entire history of appearances of a target over time. In order to reduce the dimensionality only the Eigen images of the target are computed by incremental PCA over the target's intensity values. The Last-in-First-out (LIFO) order is maintained wherein the tracker *forgets* the oldest appearance first. Foreground-Background Tracker (FBT) [49] uses a linear

discriminant classifier that is trained on Gabor texture feature vectors derived from local background, surrounding the target using color Speeded up Robust Features (SURF) [50]. Tracking by Sampling Trackers (TST) [51] relies on tracking by sampling many trackers. Each tracker is made from 4 components: a: appearance model, b: motion model c: state representation and d: an observation model. The state of the target stores the center, scale and spatial information. Multi-Domain Convolutional Neural Network Tracker (MDNet) [52] pre-trains a convolutional neural network on a generic set of sequences. During the tracking phase, samples are extracted around the target and the appearance with the maximum score is selected. Spatially Regularized Discriminative Correlation Filter with Deep Features (SRDCF) [53] uses the SRDCF tracker and uses deep features instead of a hand picked feature set. Both CNN based trackers outperformed all the other trackers and MDNet in particular is better in both accuracy and robustness than other state of the art trackers. Several trackers that participated in the challenge also compute histogram based features for matching which is quite similar to our proposed algorithm. SumShift [54] uses patch based histograms as a template descriptor. Descriptor computation is similar to LOFT with the exception of splitting up the target into pre-defined sub-windows before extracting histograms. S3Tracker is an extension of the SumShift tracker with addition of likelihood for scale of the object to be searched. Both trackers ranked in the top 15 set in VOT-2015.

Chapter 3

LOFT: Likelihood of Features Tracking System

Likelihood of Features Tracking (LOFT) system that is based on fusing multiple sources of information about the target and its environment. LOFT uses image-based feature likelihood maps derived from a template-based target model, object and motion saliency, track prediction and management, combined with a novel adaptive appearance target update model.

LOFT was primarily designed to track objects in WAMI. The overall LOFT tracking system shown in Figure 3.1, can be broadly organized into several categories including: (i) Target modeling, (ii) Likelihood fusion, and (iii) Track management. Given a target of interest, it is modeled using a rich feature set including intensity/color, edge, shape and texture information [2, 55]. The novelty of the overall LOFT system stems from a combination of critical components including a flexible set of features to model the target, an explicit appearance update scheme, adaptive

posterior likelihood fusion, a kinematic motion model, and track termination working cooperatively in balance to produce a unified reliable tracking system.

The features used in the LOFT system can be grouped into four categories: region-based, edge-based, local shape-based, and texture-based. Block-based similarity measures such as intensity and gradient cross-correlations incorporate spatial information, that histogram/distribution-based similarity measures lack and provide better discrimination power, but are sensitive to pose and viewing orientation. On the other hand histogram-based techniques provide global information about objects and image windows that are tolerant of small changes due to motion, illumination, pose, or viewing direction. We primarily use histogram-based descriptors and similarity measures except for the normalized intensity and gradient magnitude correlation to estimate feature likelihood maps. Gradient magnitude normalized cross-correlation and gradient magnitude histograms are edge-based features computed similar to their intensity counterparts. Gradient orientation information is captured using the histogram of oriented gradients (HOG) descriptor which has been successfully used in many recent object and people detection applications [56]. HOG bins the gradient magnitude weighted gradient orientations over an image patch and is a dense version of the popular scale-invariant feature transform (SIFT) descriptor. Robust orientation estimation is important for HOG-like descriptors. Our novel extension uses more accurate orientation estimation based on the adaptive robust structure tensor (ARST-HOG) [57]. Structure tensors are a useful tool for reliably estimating oriented structures within a neighborhood even in the presence of noise. In our preliminary car detection results ARST-HOG outperformed standard HOG. Local shape-based

features are measured using the eigenvalues of the Hessian matrix H, of the intensity field I(x, y), that describes the second order structure of local intensity variations around each image point, Two measures of local shape are the shape index and the normalized curvature index features derived from the eigenvalues. In the experimental results an unsigned ordering of the eigenvalues was used. A third shape measure is the magnitude weighted histogram of the Hessian eigenvector orientations. This descriptor is similar to HOG but with orientations corresponding to Hessian (Jacobian of image gradient) eigenvectors instead of a simple gradient measure. Textures are easy to recognize but hard to define. Texture analysis approaches include features of co-occurrence matrices, spatial filtering, random field models and texton pattern modeling. A simple texture measure that combines statistical and structural models of texture is based on the local binary pattern (LBP) histogram [58]. The LBP characterizes the quantized local intensity variability and various extensions to LBP have been proposed including the median binary pattern (MBP) [59]. We use the uniform rotation-invariant LBP consisting of 18 unique patterns. Feature likelihood maps are computed using sliding window histogram differencing. Local maxima in the feature likelihood maps that exceed a threshold are considered as high probability target locations.

The likelihood map of each feature is generated by comparing the feature histograms of the target and search regions. Each pixel in a likelihood map is a probability measure of the pixel belonging to the background. The likelihood maps are then fused with a weighted sum and one of the important parameters of this step is the relative importance of features. Previously we examined several different fea-

ture fusion weighting schemes including Variance Ratio [18], Distractor Index [2] and Feature Prominence [60] for fusion.

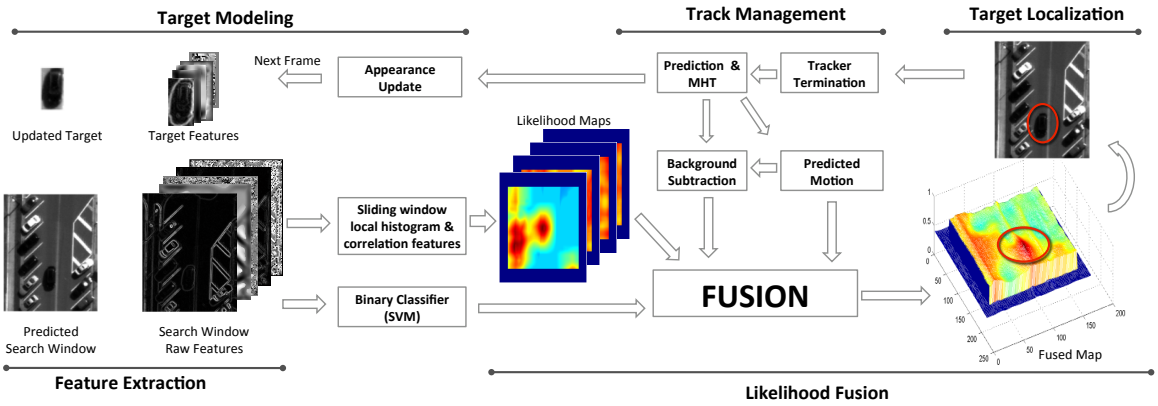


Figure 3.1: Likelihood of Features Tracking (LOFT) processing pipeline showing major components including feature extraction, feature likelihood map estimation by combining with the template, vehicle detection using support vector machine (SVM) classification, fusion module that also incorporates prediction based motion and background subtraction based motion, to produce a fused likelihood for target localization. The track management includes termination module, prediction with or without multiple hypothesis tracking (MHT) and object appearance updating for adaptive target modeling [42].

The rest of the chapter is organized as follows. Section 3.1 describes estimating features and fusing the posterior likelihood maps. Section 3.2 describes the novel target appearance modeling and adaptive update modules. Section 3.3 describes the tracker management component that is often lacking in other systems including smooth trajectory assessment and appropriate tracker termination to maintain track purity.

3.1 Likelihood Fusion

The target area is modeled using features that can be grouped into categories such as block, edge, shape and texture based. We use local binary patterns for texture, eigenvalues of the Hessian matrix for shape information, gradient orientation information using Histogram of Oriented Gradients, gradient magnitude as edge-based features and intensity as block based features. More details about these feature descriptors and their computation presented in Palaniappan et al. [2].

The tracker decides the presence of target and its location according to the likelihood of target being within the search region. A likelihood map is produced for each feature by comparing the feature histograms of target and the search regions using a sliding window-based approach (see Fig. 3.1). Each pixel in the likelihood map indicates the posterior probability of that pixel belonging to the target. Using multiple features enables adaptation of the tracker to dynamic environment changes and target appearance variabilities. It also provides more robust localization especially for cluttered environments. Feature adaptation can be accomplished using fusion. In our LOFT algorithm we use a weighted sum fusion rule which tends to perform better than other methods such as the product rule [61]. The critical aspect in weighted sum fusion is the relative importance of feature maps. Each feature performs differently depending on the target characteristic and environmental situations during tracking. Equally weighted fusion of likelihood maps can decrease performance, when some of the features are not informative in a particular environment. The importance assigned to each feature can be adapted to the changes in target pose and the surrounding background. Temporal feature weight adaptation can improve performance

under changes that are not explicitly modeled by the tracker.

We considered two weighting schemes including the Variance Ratio (VR) [18] and the Distractor Index [2]. LOFT fuses the histogram based and correlation based features in two stages. Firstly, histogram based features are fused using the VR method [18, 62] which adaptively weights the features according to the discriminative power between the target and the background measured using the two-class ratio of total to within class variances. Secondly, non-histogram (*i.e.* correlation) based features are combined with the fused histogram-based features using the Distractor Index method proposed by Palaniappan *et al.* [2]. In this method, the number of local maxima within 90% of the maximum likelihood and within the approximate spatial support of the object template, $\mathcal{N}_T$, are estimated and used as the number of viable peaks for the i^{th} feature, $m_i \in [1, \infty)$. Fusion feature weights in LOFT are then calculated using [2],

$$w_i \approx m_i^{-1}(\sum_{i=1}^{n} 1/m_i)^{-1}. \tag{3.1}$$

Consequently, high distractor index values will result in low weights for unreliable features. By assuming the environment does not change drastically across frames, the system fuses the likelihood maps of frame k using the feature weights which were estimated at frame $k-1$. Calculating feature weights dynamically enables the tracker to cope with small appearance changes in target and environment. Strong local maxima in the fused map which exceeds a predetermined threshold are considered as potential target locations.

3.2 Target Modeling

LOFT [2] uses the principle of single target template-based tracking where target features are used to match an area or region in subsequent frames. Static template-based tracking has been studied in computer vision dating back to at least the 1980's [63]. Currently, generative models such as [16, 17] or discriminative models such as [20, 64] all have online and offline versions to robustly adapt to object appearance variability. Recently, several trackers based on sparse representation have shown promise in handling complex appearance changes [24, 65, 66]. Our dynamic appearance adaptation scheme maintains and updates a single template by estimating affine changes in the target to handle orientation and scale changes [67], using multiscale Laplacian of Gaussian edge detection followed by segmentation to largely correct for drift. Multi-template extensions of the proposed approach are straightforward but computationally more expensive.

3.2.1 Appearance Update

Given a target object template, T_s, in the initial starting image frame, I_s, we want to identify the location of the template in each frame of the WAMI sequence using a likelihood matching function, $M(\cdot)$. Once the presence or absence of the target has been determined, we then need to decide whether or not to update the template. The target template needs to be updated at appropriate time points during tracking, without drifting off the target, using an update schedule which is a tradeoff between plasticity (fast template updates) and stability (slow template updates). The template search

and update model can be represented as,

$$\mathbf{x}^*_{k+1} = \arg\max_{\mathbf{x}\in\mathcal{N}_W} M(I_{k+1}(\mathbf{x}), T_u),\ k \geq s,\ u \geq s \tag{3.2}$$

$$T_{k+1} = \begin{cases} I_{k+1}(\mathbf{x}^*_{k+1} + c), & \text{if } f(\mathbf{x}^*_{k+1}, I_{k+1}, T_u)) > Th \\ \underset{c\in\mathcal{N}_T}{} & \\ T_u, & \text{otherwise} \end{cases} \tag{3.3}$$

where $M(\cdot)$ denotes the posterior likelihood estimation operator that compares the vehicle/car template from time step u, T_u (with support region, $c \in \mathcal{N}_T$), within the image search window region, $\mathcal{N}_W$, at time step $k+1$. The optimal target location in I_{k+1} is given by $\mathbf{x}^*_{k+1}$. If the car appearance is stable with respect to the last updated template, T_u, then no template update is performed. However, if the appearance change function is above a threshold indicating that the object appearance is changing and we are confident that this change is not due to an occlusion or shadow then the template is updated to the image block centered at $\mathbf{x}^*_{k+1}$. Instead of maintaining and updating a single template model of the target a collection of templates can be kept (as in learning-based methods) using the same framework, in which case we would search for the best match among all templates in Eq. 3.2. Note that if $u = s$ then the object template is never updated and remains identical to the initialized target model. Our adaptive update function $f(\cdot)$ considers a variety of factors such as orientation, illumination and scale changes.

In most video object tracking scenarios the no update scheme rarely leads to better performance [16] whereas naively updating on every frame will quickly cause the

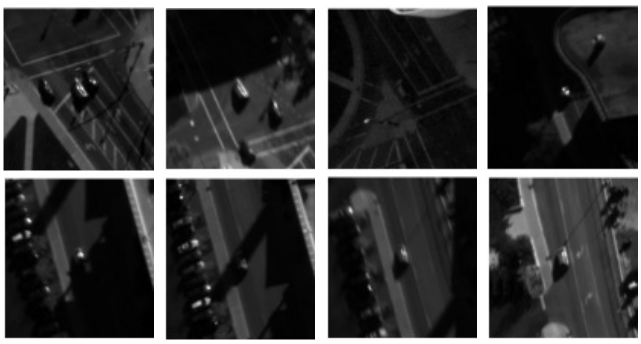


Figure 3.2: Orientation and intensity appearance changes of the same vehicle over a short period of time necessitates updates to the target template at an appropriate schedule balancing plasticity and stability [42].

tracker to drift especially in complex video such as WAMI [2]; making the tradeoff between these two extremes is commonly referred to as the stability-plasticity dilemma [68]. Figure 3.2 shows several frames of a sample car from the CLIF sequences as its appearance changes over time. Our approach to this dilemma is to explicitly model appearance variation by estimating scale and orientation changes in the target that is robust to illumination variation. Segmentation can further improve performance [69, 70].

We recover the affine transformation matrix to model the appearance update by first extracting a reliable contour of the object to be tracked using a multiscale Laplacian of Gaussian, followed by estimating the updated pose of the object using the Radon transform projections as described below.

3.2.2 Laplacian of Gaussian

A multi-scale Laplacian of Gaussian (LoG) filter is used to increase the response of the edge pixels. Using a series of convolutions with scale-normalized LoG kernels

$\sigma^2\nabla^2 G(x, y, \sigma^2)$ where σ denotes the standard deviation of the Gaussian filter,

$$I_{k,L}(x, y, \sigma^2) = I_k(x, y) * \sigma^2\nabla^2 G(x, y, \sigma^2) \tag{3.4}$$

we estimate the object scale at time k by estimating the mean of the local maxima responses in the LoG within the vehicle template region $\mathcal{N}_T$. If this $\hat{\sigma}_k^*$ has changed from $\hat{\sigma}_u^*$ then the object scale is updated.

3.2.3 Orientation Estimation

The Radon transform is used to estimate the orientation of the object [67] and applying the transform on the LoG image $I_{k,L}(x, y)$, we can denote the line integrals as:

$$R_k(\rho, \theta) = \iint I_{k,L}(x, y)\delta(\rho - x\cos\theta - y\sin\theta)\ dx\ dy. \tag{3.5}$$

where $\delta(\cdot)$ is the Dirac delta sampling function that samples the image along a ray. Given the image projection at angle θ, we estimate the variance of each projection profile and search for the maximum in the projection variances by using a second order derivative operator to achieve robustness to illumination change [71]. An example of vehicle orientation and change in orientation estimation is shown in Figure 3.3. This appearance update procedure seems to provide a balance between plasticity and stability that works well for vehicles in aerial imagery. More detailed explaination of orientation estimation is found in our related work [72] and is described in the upcoming Chapter 5.

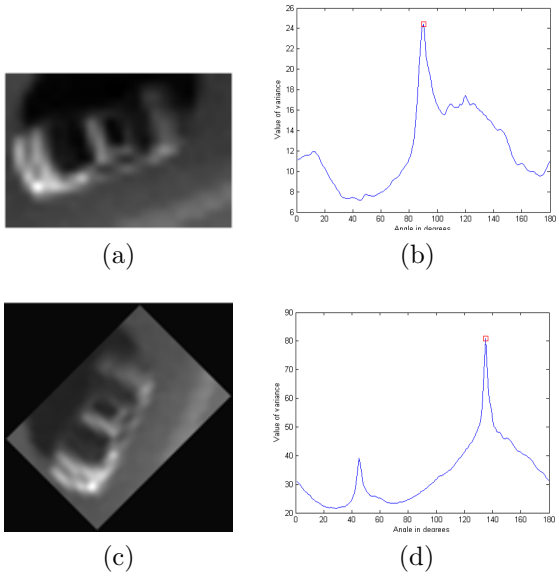


Figure 3.3: Vehicle orientations are measured wrt vertical axis pointed up. (a) Car template. (b) Variance of Radon transform profiles with maximum at 90° (red square). (c) Car template rotated by 45° CCW. (d) Peak in variance of Radon transform profiles at 135° (red square). Change in car orientation is correctly reported as 45° [42].

3.3 Track Management

A robust tracker should maintain track history information and terminate the tracker when performance is deteriorating irrecoverably (e.g. camera seam boundary), the target leaves the field-of-view (e.g. target exiting the scene), enters a long occluded/shadow region, or the tracker has lost the target. LOFT incorporates multiple *track termination* conditions to ensure high precision (track purity) and enable downstream tracklet stitching algorithms to operate efficiently during *track stitching*. Track linearity or smoothness guides the tracker to select more plausible target locations incorporating vehicle motion dynamics and a termination module for terminating the

tracker.

3.3.1 Smooth Trajectory Dynamics Assumption

Peaks in the fused likelihood map are often many due to clutter and denote possible target locations including distractors. However, only a small subset of these will satisfy the smooth motion assumption (*i. e.* linear motion). Checks for smooth motion/linearity is enforced before a candidate target location is selected to eliminate improbable locations. Figure 3.4 illustrates the linear motion constraint. The red point indicates a candidate object with a very similar appearance to the target being tracked, but this location is improbable since it does not satisfy the trajectory motion dynamics check and so the next highest peak is selected (yellow dot). This condition enforces smoothness of the trajectory thus eliminating erratic jumps and does not affect turning cars.

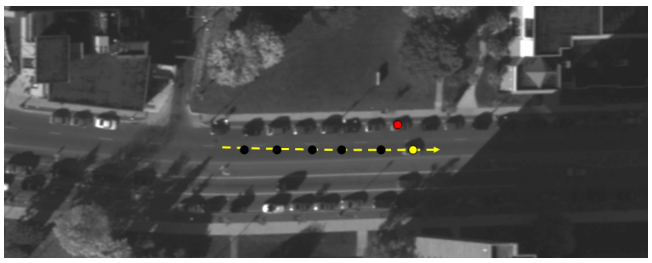


Figure 3.4: When the maximum peak (red dot) deviates from the smooth trajectory assumption (in this case linearity) LOFT ignores the distractor to select a less dominant peak satisfying the linearity constraint (yellow dot) [42].

3.3.2 Prediction & Filtering Dynamical Model

LOFT can use multiple types of filters for motion prediction. In the implementation evaluation for this paper we used a Kalman filter for smoothing and prediction [73, 74] to determine the search window in the next frame, I_{k+1}. The Kalman filter is a recursive filter that estimates the state, $\mathbf{x}_k$, of a linear dynamical system from a series of noisy measurements, $\mathbf{z}_k$. At each time step k the state transition model is applied to the state to generate the new state,

$$\mathbf{x}_{k+1} = \mathbf{F}_k \, \mathbf{x}_k + \mathbf{v}_k \tag{3.6}$$

assuming a linear additive Gaussian process noise model. The measurement equation under uncertainty generates the observed outputs from the true ("hidden") state.

$$\mathbf{z}_k = \mathbf{H}_k \, \mathbf{x}_k + \mathbf{w}_k \tag{3.7}$$

where $\mathbf{v}_k$ denotes process noise (Gaussian with zero-mean and covariance $\mathbf{Q}_k$), $\mathbf{w}_k$ denotes measurement noise (Gaussian with zero-mean and covariance $\mathbf{R}_k$). The system plant is modeled by known linear systems, where $\mathbf{F}_k$ is the state-transition matrix and $\mathbf{H}_k$ is the observation model.

Possible target locations within the search window are denoted by peak locations in the fused posterior vehicle likelihood map. Candidate locations are then filtered by incorporating the prediction information. Given a case where feature fusion indicates low probability of the target location (due to occlusions, image distortions, inadequacy of features to localize the object, etc.) the filtering-based predicted position is

then reported as the target location. Figure 3.5 shows LOFT with the appearance-based update module being active over the track segments in yellow with informative search windows, whereas in the shadow region the appearance-based features become unreliable and LOFT switches to using only filtering-based prediction mode (track segments in white).

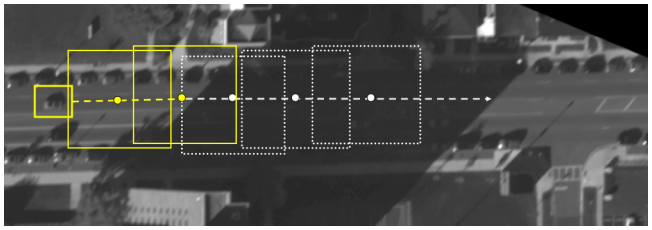


Figure 3.5: Adaptation to changing environmental situations. LOFT switches between using fused feature- and filterin-based target localization (yellow boxes) within informative search windows (yellow boxes) and predominantly filtering based localization in uninformative search windows (white boxes) [42].

3.3.3 Target vs Environment Contrast

LOFT measures the dissimilarity between the target and its surrounding environment in order to assess the presence of occlusion events. If the variance ratio between the target and its environment is below a threshold, this indicates a high probability that the tracker/target is within an occluded region. In such situations, LOFT relies more heavily on the Kalman filter. Figure 3.6 shows a sample frame which illustrates the difference between high and low VR locations.

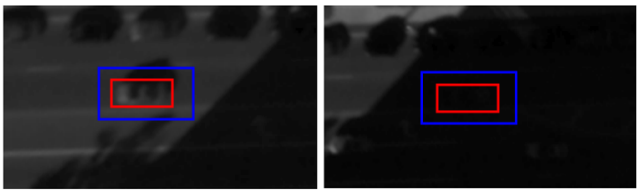


Figure 3.6: Pixels within the red rectangle form the foreground (Fg) distribution, pixels between the red and blue rectangles form the background (Bg) distribution. (Left) High VR: Fg and Bg regions have different distributions. (Right) Low VR: Fg and Bg regions have similar distributions [42].

3.3.4 Image/Camera Boundary Check

LOFT determines if the target is leaving the scene, crossing a seam or entering an image boundary region on every iteration in order to test for the disappearance of targets. If the predicted location is out of the working boundary, the tracker automatically terminates to avoid data access issues (Figure 3.7). LOFT, as a tracking

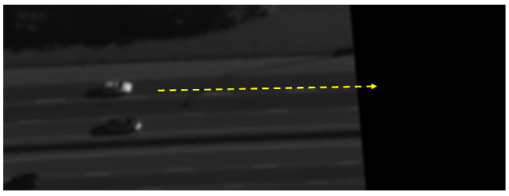


Figure 3.7: Termination of tracks for targets leaving the working image boundary [42].

system, encompasses the modules described in this section. Image boundary checks are required required for graceful termination, handling of partial and full occlusions is required for continuing tracking effectively increasing track length. The linearity check imposes constraints on smooth linear motion which is typical of objects in the real world and especially vehicles driving on paved roads. The orientation estimation modules are described more in detail in the upcoming chapters 4 and 5.

Chapter 4

Orientation Estimation Using Radon Transform

4.1 Introduction

The Radon transform in the discreet form is widely used in image processing and particularly in biomedical imaging. Magnetic Resonance Imaging and fan beam reconstructions use the inverse Radon transform to assemble the projections. The projections are also used as a feature to describe and detect lines. A discreet version of Hough and Radon transforms are related but they have key differences. The use of projections is a very powerful descriptor for lines and line detection. The accumulation of votes in the transform profile space can be used to describe lines. The general Radon transform used in tomography and microscopy where the core function $g(x, y)$

has no preferred orientation can be described by a line in its normal form:

$$\rho = xcos\theta + ysin\theta \tag{4.1}$$

The general equation is written in the following form

$$R(\rho,\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y)\delta(\rho - xcos\theta - ysin\theta)dxdy \tag{4.2}$$

4.1.1 Radon Transform of a Line

Using the normal equation of the Radon transform a line can be modeled with certain parameters (ρ^*, θ^*) as given in Peter Toft's thesis [75] is as follows:

$$\begin{aligned}
R(\rho,\theta) &= \delta(\rho^* - xcos\theta^* - ysin\theta^*) \\
\hat{R}(\rho,\theta) &= \int_{-\infty}^{\infty} \delta(\rho^* - (\rho cos\theta - ssin\theta)cos\theta^* - (\rho sin\theta + scos\theta)sin\theta^*)ds \\
&= \int_{-\infty}^{\infty} \delta(\rho^* - \rho cos(\theta - \theta^*) + ssin(\theta - \theta^*))ds \\
&= \int_{-\infty}^{\infty} \frac{1}{|sin(\theta - \theta^*)|}\delta\left(\frac{\rho^* - \rho cos(\theta - \theta^*)}{sin(\theta - \theta^*)} + s\right)ds, \qquad (4.3)
\end{aligned}$$

if $sin(\theta - \theta^*) \neq 0$

$$= \frac{1}{|sin(\theta - \theta^*)|} \tag{4.4}$$

A peak if formed when $\rho = \rho^*$ and $\theta = \theta^*$ and this basic property can be used to detect the spatial location and orientation of the lines. We have formulated our problem as detecting orientation when an input image contains a set of lines or edges.

While the technique is designed to detect lines, we have shown that using the raw output is not always ideal and needs further calculations in addition to get acceptable tolerances to orientation change error in real applications. This is demonstrated by the orientation evaluation experiment wherein different inputs and different forms of outputs are benchmarked. In this regard we have compared our technique to a similar Radon based transform called the Geometric Transform [76]. The Geometric Transform or GeT was used for appearance modeling for a parts based descriptor in Li et al. [76].

4.2 Orientation Estimation Using Radon Transform

We consider that the object to be tracked is defined by its initial appearance and associated set of feature descriptors which are provided as input to the tracking system. We denote the given sequence of images as I_k where k is the frame number. The image patch or region, centered at (x, y), representing the target object in the initial frame I_0, is denoted as $T_0(x, y)$. Let us denote a new template T_1 as the region localizing the target in I_1, and the variation between T_1 and T_0 as V_1. Appearance variations can be modeled by changes in shape and texture [27]. For our adaptive model, we assume that the variations in shape and texture can be modeled by an affine transformation matrix $\mathrm{A_k}$, that includes translation, rotation, scale and shear changes between time $k-1$ and k. When the variation in appearance V_k is significant then an appearance update is performed.

4.2.1 Orientation Estimation

We propose a robust way to estimate the orientation of vehicle objects based on the Radon transform assuming that a reliable binary map is available. The Radon transform computes a projection of the image as a sum of line integrals accumulating pixel intensities along rays at a set of angles. The Radon transform line integral, $R(\rho, \theta)$, maps the image to a new ρ and θ space, where a ray is given as, $\rho = x\cos\theta + y\sin\theta$. The Radon transform of the binarized map, $\tilde{I}_{k,H}$, at time k is defined as,

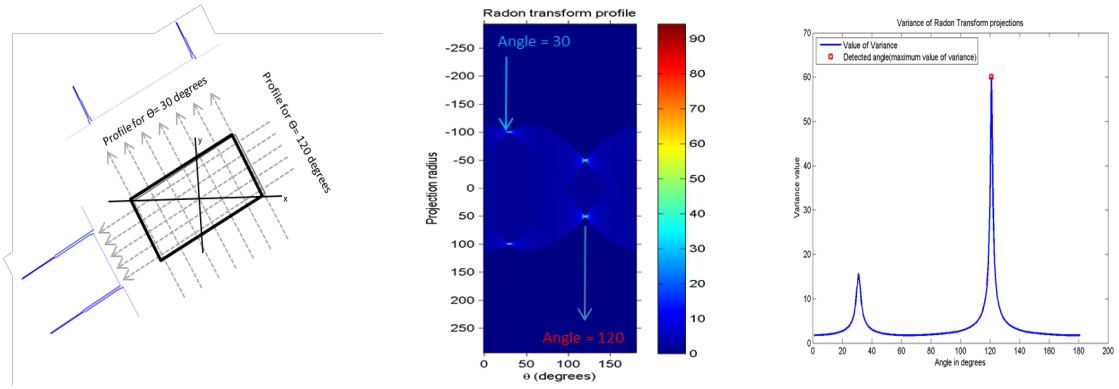
$$R_k(\rho, \theta) = \iint H(\mathcal{H}_W)\delta(\rho - x\cos\theta - y\sin\theta)dx\,dy \tag{4.5}$$

where $\delta(\cdot)$ is the Dirac delta sampling function and where $H(\cdot)$ is the Heaviside function such that positive values (inclusive of zero) are binarized to one and negative values as zero. Given the Radon transform projections we can calculate the variance of each profile (ρ varies while θ remains fixed),

$$V_k(\theta_F) \equiv Var(R_k(\rho, \theta = \theta_F)) \tag{4.6}$$

The second derivative properties of the profile variance function, $V_k(\theta)$, are quite robust as an estimate of local structure orientation [71]. However, for our purposes looking at the maximum value in the variance function results in sufficient accuracy in the estimated angle. Figure 4.1 shows the example of a simple shape and it's corresponding result with the proposed method. The simple rectangle demonstrates how the cumulative sum affects the radon transform profile. The shorter sides and the longer sides show up as peaks and the magnitude difference shows how the longest

edge is picked as being the orientation of the object.



(a) Projections of binary image (b) Radon Transform (RT) Profile (c) Variance of RT profile

Figure 4.1: Diagram showing Radon Transform projections on a binary image and the resulting profile and variance curve. The estimated angle is at 120 degrees (measured w.r.t vertical axis) [77]

4.2.2 Similarity to Geometric Transform

Modeling appearance inside a closed contour was proposed by Li et al. [76]. This general transform was designed to combine shape and appearance information at different resolutions and be invariant to deformations and occlusions. The different formulations that were suggested were to model and encode the appearance information. While not being explicitly used for orientation estimation, a similar procedure to our Radon transform based estimation approach can be used to determine appearance orientation. Since the transform is designed to encode changes in the appearance of the object we believe that with similar techniques to estimate orientation we have another very closely related method to compare against. We use the GeT equation 14

from [76] as it is the closest in its formulation to our direct Radon transform technique defined as follows:

$$R(\rho,\theta) = \frac{\int g(x,y)\delta(xcos\theta + ysin\theta - \rho)dxdy}{\int H(I(x,y))\delta(xcos\theta + ysin\theta - \rho)dxdy} \tag{4.7}$$

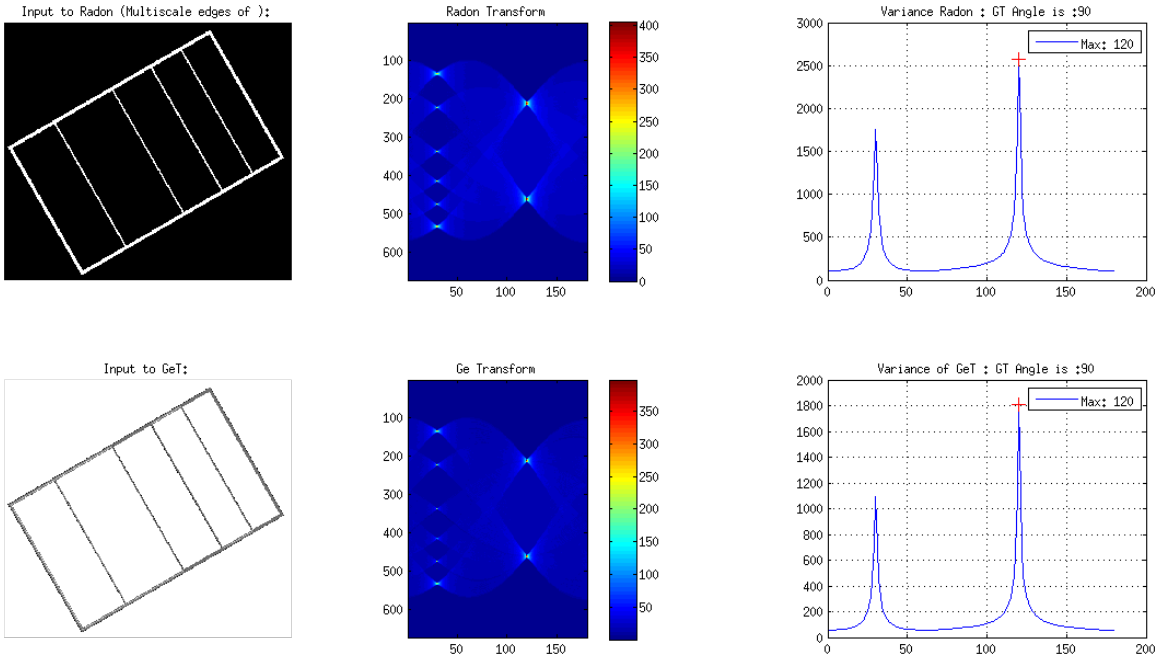


Figure 4.2: Figure showing Radon Transform projections on a binary image and the resulting profile with its corresponding variance curve. The estimated angle is at 120 degrees (measured w.r.t vertical axis). The bottom row shows the input to the Geometric Transform (GeT) and its corresponding profile and variance plot. In this case we are simulating an ideal segmentation groundtruth for the input to GeT by providing a binary mask with ones only where there are gray values other than zero. In this case, the GeT and the Radon Transform would determine the same angle but there are key differences like the magnitude of the profile plot (y-axis) and the multiple peak values (y-axis) in the variance plots.

In the rest of the chapter, we have compared the performance of both the techniques. As an observation, the GeT simply uses only the projections that are con-

tained inside a masked region. Our experiments show that given an ideal segmentation the GeT performs exactly the same as our approach of using binarized multiscale edges and the Radon transform. However, the performance also relies mostly on the quality of the segmentation which in our approach is not much of a problem at the stage of computing the orientation. Figure 4.2 shows an ideal segmentation case for GeT and perfect binary edges for input to our Radon transform approach.

4.2.3 Evaluation: Radon versus GeT

Table 4.1 shows a comprehensive evaluation between GeT and our Radon method with different inputs. These experiments were conducted on Columbus Large Image Format(CLIF) [78] dataset using groundtruth bounding boxes as input. The detailed description of the CLIF data is given in Chapter 7. The use of groundtruth bounding boxes eliminates inaccuracies in tracking and since this evaluation is focused only on estimating the orientation correctly, the assumption of having ideal input is important to highlight key differences. The groundtruth boxes have orientation information since they are manually marked polygons instead of axis aligned boxes. We then compare the estimated orientation against the orientation from the groundtruth and if the estimate deviates by a certain number of degrees (in this case 5 degrees) then it is considered as a wrong estimate for that frame. Maintaining a counter for the number of correct estimates Table 4.1 shows this number for the different stated techniques. Peak (P) indicates the angle associated with the maximum value in the 2D Radon Transform of the associated feature map (ie intensity, Hessian, or binarized Hessian) is used as the estimate of the vehicle orientation. Variance (V) shows results

	Radon								*GeT(G)*							
	Radon(R)															
	Intensity(I)			*Hessian(H)*					*Intensity(I)*			*Hessian(H)*				
	Raw			*Raw*			*Binary(B)*		*Raw*			*Raw*			*Binary(B)*	
	Peak (P)	Variance (V)	Variance (V)	Peak (P)	Peak (P)	Variance (V)		Peak (P)	Variance (V)	Peak (P)		Variance (V)	Peak (P)	Variance (V)		
	Peak (P)	Variance (V)		Peak (P)	Variance (V)			Peak (P)	Variance (V)			Peak (P)	Variance (V)			
C0_3_0	17	4		28	28	33	**28**		22	11		13	5	0	12	
C1_2_0	10	15		10	21	25	**25**		14	9		9	13	0	16	
C1_4_0	36	18		43	36	31	**27**		13	12		12	3	0	14	
C1_4_6	36	38		20	24	27	**23**		11	7		10	9	2	13	
C2_4_1	21	25		19	28	19	**26**		6	5		4	3	0	20	
C3_3_4	7	13		13	17	6	**11**		6	6		1	1	6	9	
C4_1_0	9	9		6	6	7	**7**		6	1		5	1	6	7	
C4_3_0	4	4		16	8	15	**17**		8	4		0	0	0	13	
C4_4_1	17	16		16	16	23	**25**		18	21		13	27	0	14	
C4_4_4	12	9		12	12	10	**11**		0	0		0	0	0	5	
C5_1_4	18	5		23	24	24	**24**		0	0		4	0	8	22	
C5_2_0	4	14		27	30	30	**29**		19	24		3	0	32	28	
C5_3_7	27	27		27	27	27	**27**		27	5		4	1	0	26	
C5_4_1	17	5		19	19	19	**19**		13	10		6	1	0	16	
Total TP	235	202		279	296	296	**299**		163	115		84	64	54	215	

Table 4.1: CLIF Orientation evaluation on all 14 sequences. The numbers indicate the number of frames per sequence which are considered as true positive (TP) where if the angle estimate by a particular method (columns) is within 5 degrees as when compared with groundtruth angle. Hierarchical arrangement of 6 combinations for each method such that binary code is ordered from top to bottom. For e.g. RHBV $\rightarrow$ Radon Binarized Hessian Variance. **Bold** denotes our method

where the predicted angle is estimated by maximizing the variance that is in turn derived from the transform profile. There are three different types of image or 2D map inputs to compute the transforms such as intensity, hessian (H) and binarized hessian (BH). The results as shown in Table 4.1 indicate that our technique with the binarized hessian and maximized variance has a minor improvement over maximizing for the angle over the Radon transform profile. While RHBV, would be better than the other methods, we need to design an experiment that could closely reflect the accuracy difference better.

In this chapter we have seen the formulation of our proposed algorithm where we use a binary map as input to the Radon transform. Once we compute the Radon transform profile, the next step is to compute the variance which would then, once maximized, indicate the predominant orientation in the map. The method is robust

and shows promise as compared with another similar transform (GeT). In the subsequent chapter we will study how these equations are used in appearance modeling.

Chapter 5

Appearance Modeling and Adaptive Template Update Scheme

Visual feature-based tracking systems need to adapt to variations in the appearance of an object and in the scene for robust performance. Though these variations may be small for short time steps, they can accumulate over time and deteriorate the quality of the matching process across longer intervals. Tracking in aerial imagery can be challenging as viewing geometry, calibration inaccuracies, complex flight paths and background changes combined with illumination changes, and occlusions can result in rapid appearance change of objects. Balancing appearance adaptation with stability to avoid tracking non-target objects can lead to longer tracks which is an indicator of tracker robustness. The approach described in this paper can handle affine changes such as rotation by explicit orientation estimation, scale changes by using a multiscale Hessian edge detector and drift correction by using segmentation. We propose an appearance update approach that handles the 'drifting' problem using this adaptive

scheme within a tracking environment that is comprised of a rich feature set and a motion model.

Object tracking is an important task from the point of view of security and surveillance, recognizing people and objects of interest and is important towards other related tasks such as automatic annotation and retrieval, human computer interaction. Monitoring traffic patterns though related to surveillance has in recent years enveloped more complex techniques like higher level fusion of other information such as persistent tracking, association of entities and events and behavioral information. Tracking can be challenging due to noise content in the images, articulated motion, partial and full occlusions, illumination changes. Exclusive to wide-area motion imagery are other complications like camera geometry errors, smooth change in camera angles coupled with ever changing illumination creates a unique combination which makes the task challenging. Our tracking method is based on fusing multiple features by comparing a target appearance model within a region on interest using feature likelihood maps which estimates the likelihood and thus detects the most probable location of the object. Appearance modeling is a complex problem and many techniques in the literature exist that address this problem [17, 27, 79, 80]. Our main contributions are: (1) we show that appearance based trackers can result in reliable trajectories if it has a robust appearance and motion model, (2) we show that adapting to orientation of the objects and correcting for the drift can result in increased accuracy without significant overhead. Our CLIF tracking results in this paper are significantly improved over our previous results [72] due to the following extensions to LOFT with orientation estimation: (a) contrast stretching, (b) initial template

segmentation to get a reasonably tight bounding box, (c) identifying and skipping duplicate frames using image differencing and use only the Kalman filter motion without any appearance - treat it like an occlusion, (d) more sophisticated switching between Radon transform estimation of orientation and Kalman filter estimate of motion direction instead of just Radon transform-based estimate.

5.1 LOFT: A Wide-Area Tracking System

The wide-area tracking system, LOFT [2, 42], LOFT robustly tracks vehicles in wide-area large format (WALF) video that is airborne imagery characterized by large spatial coverage, high resolution of about 25 cm GSD (Ground Sampling Distance) and low frame rate of a few frames per second. Wide-area large format imagery is also known by several other terms including wide-area aerial surveillance (WAAS), wide-area persistent surveillance (WAPS), Large Volume Streaming Data (LVSD) and wide-area motion imagery (WAMI) [1–3, 5] as described in Chapter 3 Appearance modeling or adapting to changes in object appearance has been handled in different ways in the past, either by using target observations [16, 17], fusing multiple sources of sensor information [70, 81], parts-based deformable templates [22, 82], as a learning problem using on-line boosting [64], or on-line multiple instance boosting [20], while earlier techniques used off-line classifier training [48, 83]. Off-line learning requires training on a set of image patches which introduces more complexity as the accuracy then depends on factors such as the amount of variation in the image patches, the training method used, size of the training set and avoiding over-training.

Recently, sparse representation for object classification based tracking methods are actively being investigated [24, 66], but L_1 minimization is computationally expensive, especially when very high dimensional feature vectors are used and appearance adaptation is still challenging even using sparse models. A typical assumption in appearance-based tracking is that the object's visual representation does not change significantly across a sequence of frames. Although this assumption may hold over short intervals for high frame rate full motion video, it is often not valid across longer time intervals, lower frame rates and abrupt changes in the environment or imaging conditions. One (naive) approach to accommodate appearance changes, is to update the target appearance model on every frame based on the results from the matching algorithm. Updating on every frame can lead to instabilities in representing the target due to variations in scene conditions and partial occlusions which leads to the *drifting* problem, while updating less frequently can result in the model missing appearance changes that are important for continuous tracking. Making this trade-off is referred to as the stability-plasticity dilemma [64, 68]. The following sections describe the appearance modeling strategy in detail along with algorithms that are part of the LOFT system.

5.2 Appearance Based Target Model Update Strategy

The way our proposed approach handles this dilemma is by trying to balance between naive updating on every frame and a no update scheme using a single fixed template,

by estimating when there is significant change in the appearance prior to an update. Model updates can occur when there is rapid change in pose of the object or illumination change thus maintaining *plasticity*,on the other hand when the changes in orientation, scale or illumination is very small then an update is not performed and contributes to model *stability*. Cars on turns contribute to significant change in template appearance. We model this variation by estimating the change in orientation from one frame to the other and taking the decision on whether the template needs an update. Making the feature descriptors rotationally invariant may impact matching performance as typically in WAMI, objects only have a small gamut of appearances. WAMI data is also typically low frame rate compared to standard video sequences but for aerial surveillance $1 - 4$ frames per second is sufficient to predict change in orientation of an object. We believe that our model of estimating orientation and requesting updates is computationally inexpensive and this contributes a lot towards accuracy of the tracker. Appearance update strategies have a significant influence on the overall quality of the tracker by striving for high precision while increasing recall which is a challenging tradeoff to achieve [68]. We maintain several update strategies and select between them based on context and cues about the environment as well as target behavior. The update strategy presented here tries to address the *stability and plasticity* dilemma by employing the use of information such as amount of change in the pose and scale of the object, any illumination changes, and time interval since

the last update. Using the likelihood-based Eq. 5.2, one update condition,

$$T_{k+\triangle k} = \begin{cases} \text{Extract}(W_{k+\triangle k}, (x_{k+\triangle k}, y_{k+\triangle k})), \\ T_k, \quad \text{if angleDiff} \notin [A_L, A_U] \end{cases} \tag{5.1}$$

checks *angleDiff*, the difference in orientation between current and previous vehicle templates compared to a user specified angle range. The function Extract$(\cdot)$ returns an image patch centered at the given coordinates. Algorithm 1 describes the wrapper code that takes two templates as input and outputs the orientation. Algorithm 2 is the detailed pseudo-code that computes the orientation, the difference in orientation as compared with last known good appearance and a confidence measure. Finally, Algorithm 3 describes the template replace procedure under certain criteria once compared with thresholds.

Algorithm 1 Appearance update driver

Input: $T_{k-\triangle k}, T_k$
//Previously known good and current template (appearance)
x_k, y_k //Centroid from Likelihood function as in Eq. 5.2
W_k //Search Window of current frame
Output: $angle_k$
//Orientation angle of template at time k
$appConf$
//Radon transform similarity between $T_{k-\triangle k}$ & T_k
$Tpadded_k = Extract(W_k, (x_k, y_k), \tau_1)$
//where Extract$(\cdot)$ is an overloaded function similar to Eq. 5.1 and τ_1 is a scalar value denoting number of pixels to pad around the center
$BI_{(k)\mathcal{H}} = Canny(I_{(k)\mathcal{H}}, \tau_2)$
$BI_{(k-\triangle k)\mathcal{H}} = Canny(I_{(k-\triangle k)\mathcal{H}}, \tau_2)$
//Binarize the Frobenius norm using the Canny edge detector with threshold τ_2
$(appConf, angle_k) = RadonAngle(BI_{(k-\triangle k)\mathcal{H}}, BI_{(k)\mathcal{H}})$
//Call function RadonAngle$(\cdot)$ as described in Algorithm 2

Algorithm 2 RadonAngle

Input: $BI_{(k)\mathcal{H}}$,$BI_{(k-\triangle k)\mathcal{H}}$
//Edge maps
Output: $angle_k$ //Orientation of $BI_{(k)\mathcal{H}}$
$appConf$
//Radon transform similarity between $BI_{(k)\mathcal{H}}$ & $BI_{(k-\triangle k)\mathcal{H}}$
$angleDiff$ //Angular difference between $angle_k$ & $angle_{k-\triangle k}$
$R_k(\rho,\theta) = Radon(BI_{(k)\mathcal{H}}, \tau_3)$
$R_{k-\triangle k}(\rho,\theta) = Radon(BI_{(k-\triangle k)\mathcal{H}}, \tau_3)$
//Compute the Radon Transform as shown in Eq. 5.8
$V_k = ComputeVariance(R_k)$
$V_{k-\triangle k} = ComputeVariance(R_{k-\triangle k})$
//Compute Variance as given in Eq. 5.9
$angle_k = \underset{\theta}{\text{argmax}} V_k(\theta_F)$
$angle_{k-\triangle} = \underset{\theta}{\text{argmax}} V_{k-\triangle k}(\theta_F)$
//Find the peak in the variance by finding the maxima and return the corresponding angle
$angleDiff = Diff(angle_k, angle_{k-\triangle k})$
//Find the angular difference in degrees
$R_{(k)aligned} = CircShift(R_k, -angle_k)$
//Circular shift the Radon profile from timestep k to normalize to 'zero' degrees
$R_{(k-\triangle k)aligned} = CircShift(R_{k-\triangle k}, -angle_{k-\triangle k})$
$R = RMSE(R_{(k)aligned}, R_{(k-\triangle k)aligned})$
//Compute the RMSE between two normalized Radon maps Alg. 4
$appConf = ComputeConf(D, \tau_5)$
//Compute the confidence value based on the vector of distances and a maximum threshold as shown in Eq. 5.10

5.3 Orientation Estimation Using Radon Transform

We consider that the object to be tracked is defined by its initial appearance and associated set of feature descriptors which are provided as input to the tracking system. We denote the given sequence of images as I_k where k is the frame number.

Algorithm 3 Appearance Update Decision

Input: $Angle_{T_{k-\triangle k}}, Angle_{T_k}$ //Car orientation [0,180] degrees
$threshDiffAngleU, threshDiffAngleL$
//Allowed Upper and Lower angular difference in degrees
$appConfThresh$ //Appearance confidence threshold
$initTurn$ //Flag to denote target initialized on a turn
$updateAppStatus$ //Check frame count since last update
$appConf$ //Object appearance confidence
Output: $updateAppearance$ //Appearance update flag
$angleDiff \leftarrow |Angle_{T_{k-\triangle k}} - Angle_{T_k}|$ **mod** 180
$updateAppearance \leftarrow 0$
if $initTurn$ AND $appConfidence \geq appConfThresh$ OR $updateAppStatus$ AND $appConf \geq appConfThresh$ **then**
$updateAppearance \leftarrow 1$
end if
if $angleDiff \geq threshDiffAngleL$ AND $angleDiff \leq threshDiffAngleU$ AND $appConf \geq appConfThresh$ **then**
$updateAppearance \leftarrow 1$
end if

The image patch or region, centered at (x, y), representing the target object in the initial frame I_0, is denoted as $T_0(x, y)$. Let us denote a new template T_1 as the region localizing the target in I_1, and the variation between T_1 and T_0 as V_1. Appearance variations can be modeled by changes in shape and texture [27]. For our adaptive model, we assume that the variations in shape and texture can be modeled by an affine transformation matrix $\mathrm{A_k}$, that includes translation, rotation, scale and shear changes between time $k-1$ and k. When the variation in appearance V_k is significant then an appearance update is performed.The following subsections describe the appearance modeling approach that we use along with the criteria for updating followed by a description of the update strategies.

5.3.1 Extracting an Approximate Template

We assume that the tracking position localization is satisfactory so that we can extract an image patch with the same parameters as the initialized template model or the previously updated appearance model. The localization using the fused feature conditional likelihood map of the target is given by [42],

$$(x_k, y_k) = \arg\max_{x,y} \; L(W_k(x, y), T_{k-\triangle k}) \tag{5.2}$$

where $L(I_{\mathrm{k+1}}, T_{\mathrm{k}})$ is the likelihood matching function, $(x_{\mathrm{k+1}}, y_{\mathrm{k+1}})$ is the estimated target position at time $k+1$, and at this location we extract a patch $T_{k+1}(x_{k+1}, y_{k+1})$ that becomes the object template at time $k+1$. Let us denote W as the search window which is a sub-region of the full resolution image on which the likelihood function L, as given above, is computed. The two target appearance models, T_{k} and $T_{\mathrm{k+1}}$, with coordinate locations are used in the update module.

5.3.2 Multiscale Edge Detection

Diminishing the influence of background pixels is important as the accuracy of orientation estimation depends on a reliable edge map. A multiscale edge detection approach is used based on [84] which developed a vessel enhancement filter based on an eigenvalue analysis of the Hessian matrix across scale space. The Hessian matrix

uses second-order image derivatives:

$$\mathcal{H}_W = W_{xx} = \sigma^2 \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix} \tag{5.3}$$

where σ^2 is used to achieve invariance under image rescaling [79, 85]. Let λ_1, λ_2 be the eigenvalues and $\mathbf{e}_1$, $\mathbf{e}_2$ the corresponding eigenvectors of the Hessian matrix with $|\lambda_1| \geq |\lambda_2|$. The Hessian is computed after convolving the image with a 2D isotropic Gaussian,

$$G(x, y, \sigma_i) = (1/\sqrt{2\pi}\sigma_i) exp(-(x^2 + y^2)/(2\sigma_i^2)) \tag{5.4}$$

where σ_i is the standard deviation of the Gaussian distribution for the i^{th} scale. We search across a range of scales based on target size and use the Frobenius norm of the Hessian matrix at the optimal scale as a measure of the second order structureness, optimal scale being the scale with the maximum response in terms of the largest eigenvalue:

$$\sigma_{max}(x, y) = \underset{\sigma}{\operatorname{argmax}} |\lambda_1(x, y, \sigma)| \tag{5.5}$$

$$I_{k,\mathcal{H}}(x, y) = ||\mathcal{H}||_F \tag{5.6}$$

$$= \sqrt{\lambda_1(x, y, \sigma_{max})^2 + \lambda_2(x, y, \sigma_{max})^2} \tag{5.7}$$

The best scale at each pixel, $\sigma_{max}(x, y)$, is determined by the maximum response of the maximum eigenvalue or λ_1. A range of smaller scales is selected in order to suppress blob-like responses which after thresholding will produce an enhanced edge map while small scale responses in the background are diminished using the second

order structureness [79].

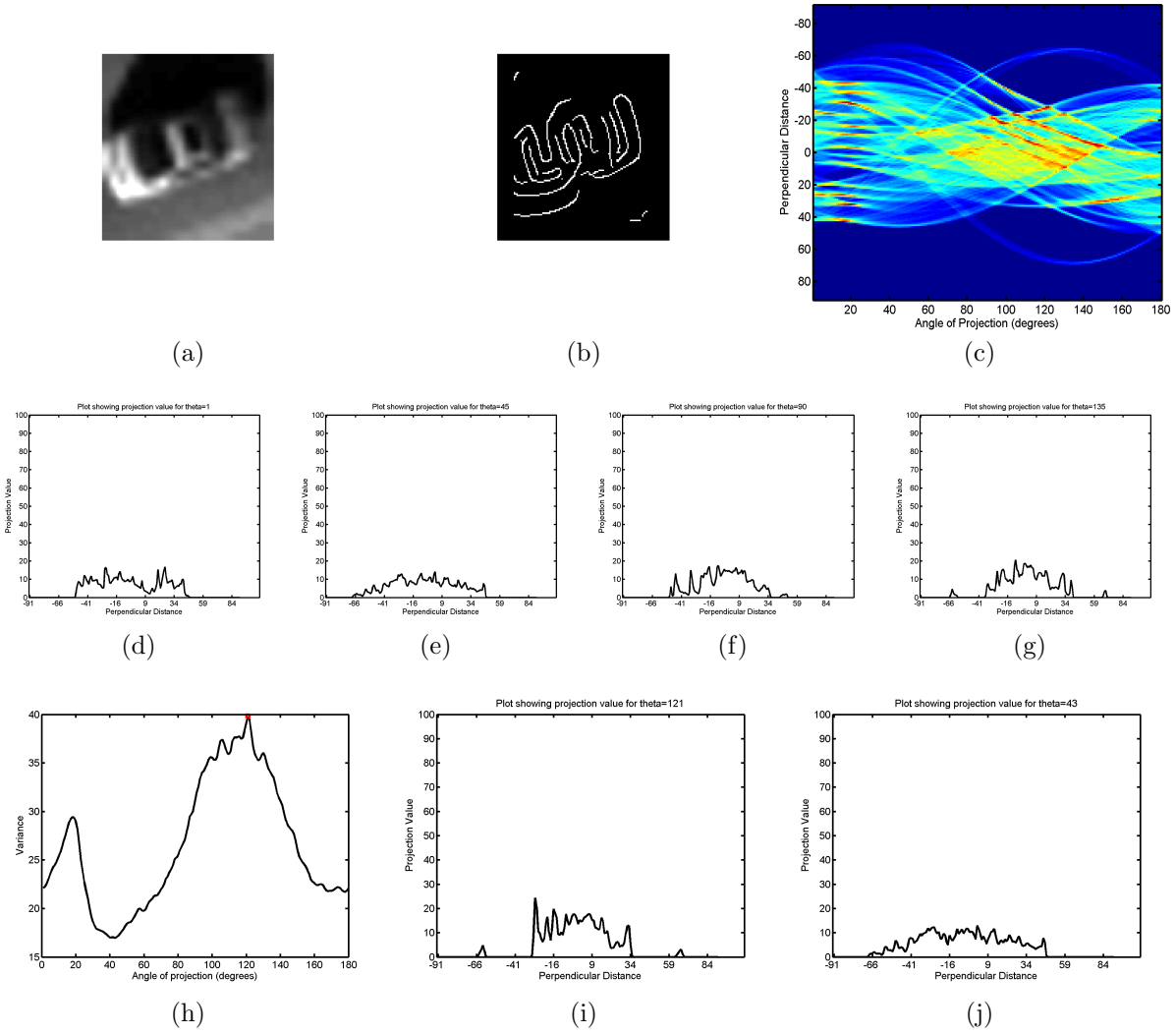


Figure 5.1: Figure showing car 1 and the corresponding orientation response. Top row shows the original image, binarized edge map and resulting Radon transform. Middle row shows the Radon transform profiles for $theta = 1, 45, 90, 135$. Last row shows the variance plot with a red marker indicating the estimated orientation and the Radon transform profiles for $theta = 121$ maximum and $theta = 43$ minimum [77].

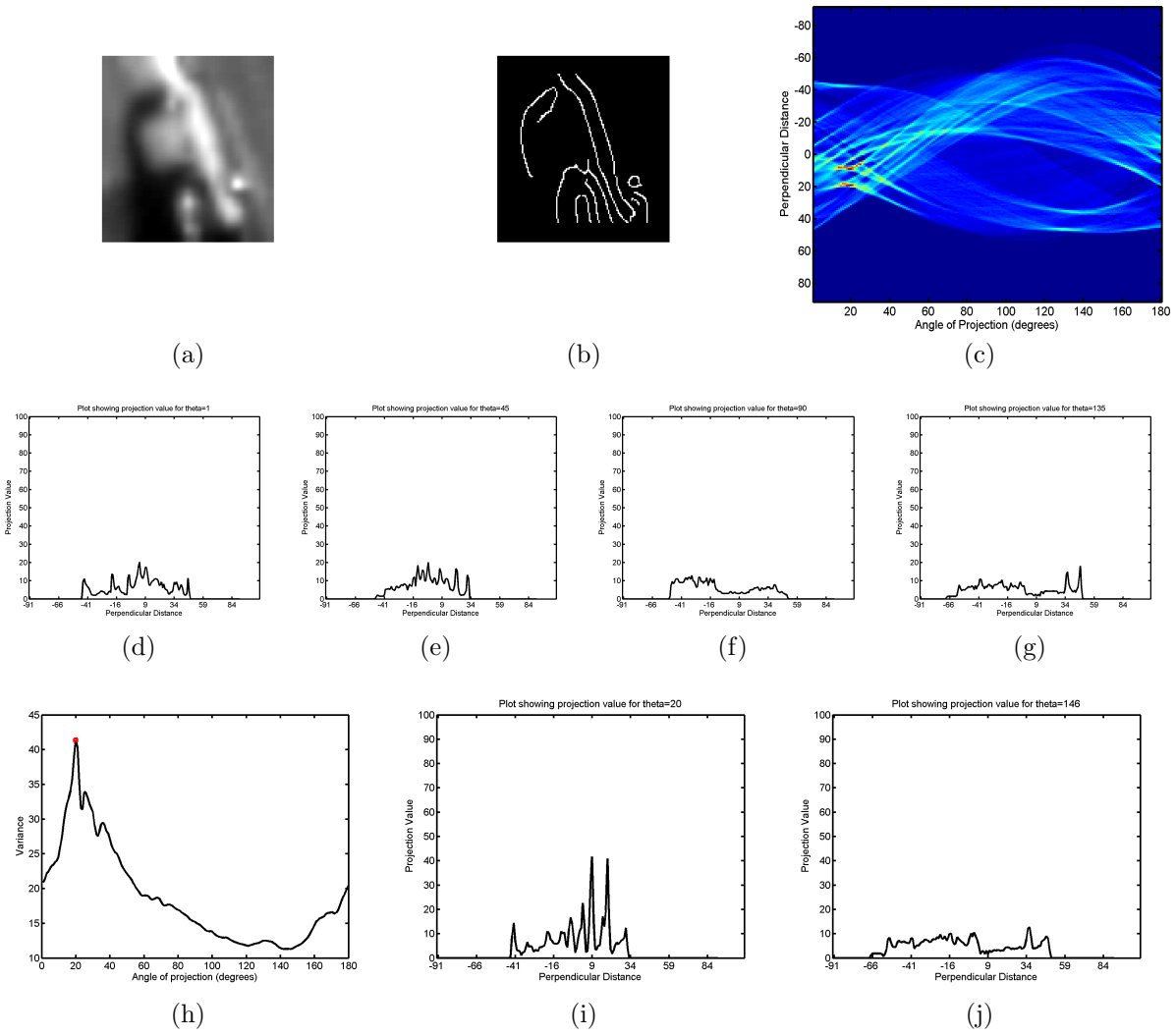


Figure 5.2: Figure showing car 14 and the corresponding orientation response. Top row shows the original image, binarized edge map and resulting Radon transform. Middle row shows the Radon transform profiles for $theta = 1, 45, 90, 135$. Last row shows the variance plot with a red marker indicating the estimated orientation and the Radon transform profiles for $theta = 20$ maximum and $theta = 145$ minimum [77].

5.3.3 Orientation Estimation

The Radon transform is used to determine the orientation as described in Chapter 4 Equation 4.5. The only difference between Equation 4.5 in Chapter 4 and Equation 5.8 is that the binarized map is the second order structureness image $I_{k,H}$ from Equation 5.5. We follow the same procedure where we compute the variance 5.9 similar to Chapter 4 Equation 4.6.

$$R_k(\rho, \theta) = \iint H(\mathcal{H}_W)\delta(\rho - x\cos\theta - y\sin\theta)dx\,dy \tag{5.8}$$

where $\delta(\cdot)$ is the Dirac delta sampling function and where $H(\cdot)$ is the Heaviside function such that positive values (inclusive of zero) are binarized to one and negative values as zero. Given the Radon transform projections we can calculate the variance of each profile (ρ varies while θ remains fixed),

$$V_k(\theta_F) \equiv Var(R_k(\rho, \theta = \theta_F)) \tag{5.9}$$

Figures 5.1, 5.2, 5.3, 5.4 show real examples from the CLIF [78] dataset where each step of the computation output is shown. The figures contain the binarized map, the radon transform profile and several profiles for a standard set of angles along with the variance plots. This shows the inner workings of the proposed method.

5.3.4 Appearance Update Confidence Using Radon Transform Similarity

A confidence value is used during the decision process to adaptively determine when an appearance update should take place to switch to the new template model $T_{\mathrm{k+i}}$ and replace T_{k}. The confidence value is adaptively estimated by selecting and associating the highest peaks in the Radon transform function (Eq. 4.5) across two time steps. Let us characterize the set of strongest peaks (strength and angle θ ignoring ρ) at two time points by lists $\mathbf{p}_{(k+i)}$ and $\mathbf{p}_k$. Algorithm 4 is used to establish peak-to-peak correspondences among the set of strongest peaks in the Radon transform domain.

Algorithm 4 RMSE of two aligned Radon Transform maps

Input: $R_{(k)aligned}, R_{(k-\triangle k)aligned)}$ // Two aligned Radon Transform maps
Output: $\mathbf{E}rr$ // RMSE of the two maps
$error = \sum((R_{(k)aligned} - R_{(k-\triangle k)aligned)})^2)/length(R_{(k)aligned})$
Err $= \sqrt{error}$

Using the list of distances $\mathbf{D}$ from Algorithm 4 we sort and find the best association of each peak in $\mathbf{p}_k$ with a single peak in $\mathbf{p}_{(k-\triangle k)}$. The resulting set of peak matching distances characterizes appearance confidence as,

$$Conf = \begin{cases} Distance/threshold, & \text{if } Distance > 0 \\ 1 - abs(Distance)/threshold, & \text{otherwise} \end{cases} \tag{5.10}$$

where D_{mean} is the mean value in the distance array $\mathbf{D}$ and represents the best association among the set of strongest peaks that in the ideal case would be zero yielding the highest confidence value of one. The distance array $\mathbf{D}$ represents how close one peak is in terms of its magnitude and orientation.

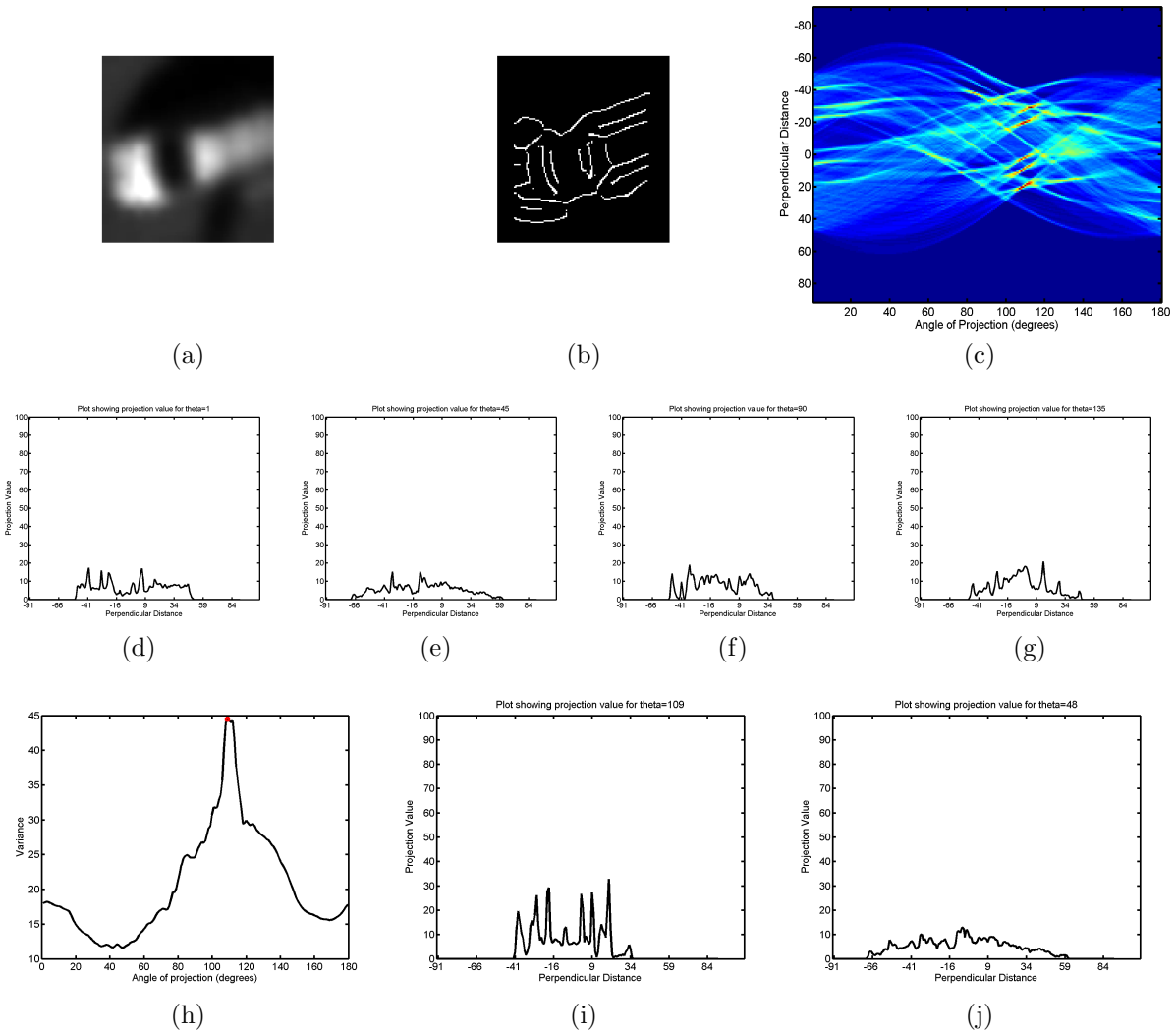


Figure 5.3: Figure showing car 10 and the corresponding orientation response. Top row shows the original image, binarized edge map and resulting Radon transform. Middle row shows the Radon transform profiles for $theta = 1, 45, 90, 135$. Last row shows the variance plot with a red marker indicating the estimated orientation and the Radon transform profiles for $theta = 109$ maximum and $theta = 48$ minimum [77].

In this chapter the LOFT appearance modeling algorithm was described in detail.

The algorithms for appearance modeling are centered around the Radon transform

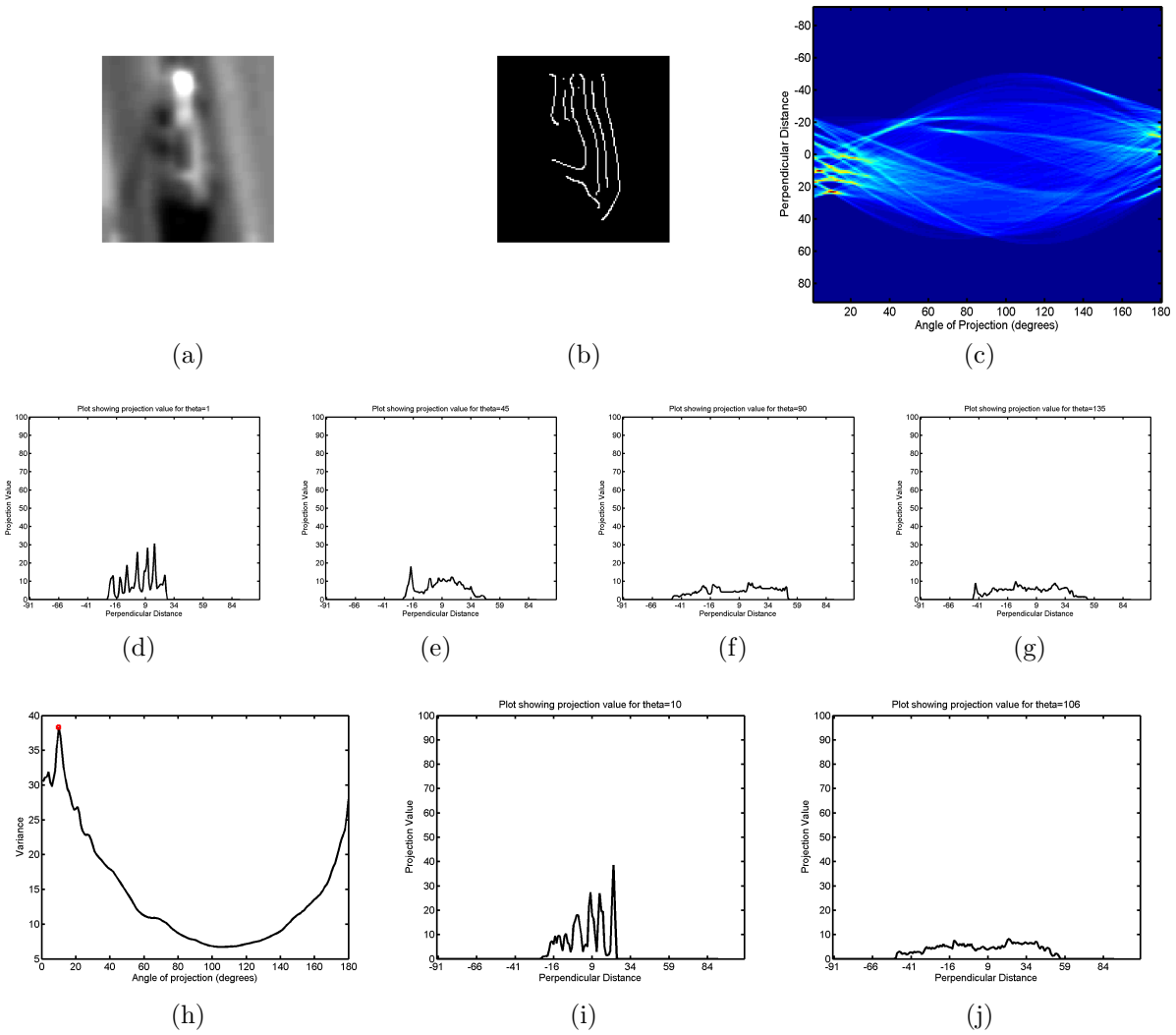


Figure 5.4: Figure showing car 12 and the corresponding orientation response. Top row shows the original image, binarized edge map and resulting Radon transform. Middle row shows the Radon transform profiles for $theta = 1, 45, 90, 135$. Last row shows the variance plot with a red marker indicating the estimated orientation and the Radon transform profiles for $theta = 10$ maximum and $theta = 105$ minimum [77].

technique described and evaluated in Chapter 4. The appearance confidence values are also derived from the Radon transform which is used to recommend an update.

Chapter 6

Applications and Implementation

The LOFT system that was described in previous chapters provides a solution to the appearance based persistent tracking. However, the implementation of such a system required a framework that would be flexible enough to be integrated with various academic and government institutions and programs. LOFT's initial implementation was a prototype in MATLAB®. The MATLAB implementation was required to be compiled using the proprietary MATLAB compiler and needed the MATLAB Compiler Runtime (MCR) during execution. Because of the MCR requirement, a lot of integration effort focused on accommodating the closed source nature of MATLAB and writing workarounds for certain shortcomings. One particular difficulty was the inability to call a MATLAB compiled library or executable in a multi-threaded way without using multiple instances of the MCR. Every instance of an MCR initialization takes about 300 MB. This would have been highly impractical to track on $100 - 200$ objects in parallel as memory requirement would increase linearly in terms of number

of objects. The effort continued to maintain a working version in MATLAB and supporting requests for integration in a single threaded environment while focusing on a pure C++ implementation. Towards the end of the integration effort we ported over all the MATLAB functionality and replaced all our ongoing code integration efforts with the native C++ implementation. As part of this, LOFT was designed and written in a fully flexible C++ framework. Functionally modular, LOFT's C++ implementation is divided into three components such as

- Driver
- Bridge and
- Core.

This implementation is shared across various projects such as Air Force Research Lab's (AFRL) Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance (C4ISR) Enterprise to the Edge (CETE) project which features an interface with Kitware's motion based tracking pipeline, AFRL's Enhanced Exploitation and Analysis Tools (E2AT), Army Video Analytics Architecture (AVAA) & a cloud based implementation separating functionality and I/O for processing in a cloud-fog environment. The following content in this chapter will describe briefly the shared codebase and the minor changes in the interface for the various projects.

6.1 On-board Realtime Target Tracking

AFRL's Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance (C4ISR) Enterprise to the Edge (CETE) program was designed to push ISR closer to the sensor. As part of this goal our integration with Kitware's KWIVER based tracking platform, we split up our tasks in two main code roll-out phases. Version one was a MATLAB® based implementation that was used to evaluate using the tracker in unassisted mode. In assisted mode the tracker is restarted as soon as it deviates a certain number of pixels away from the true position. In unassisted mode the tracker is set to run until it decides to terminate and returns control to the parent calling function. Thus, in unassisted mode the tracker needs auto termination logic which impacts true positives and false positives. Version 2 was a direct integration with the KWIVER tracking platform as a compiled MATLAB library. For invoking such code a MATLAB Compiler Runtime (MCR) which contains all the run-time libraries required to run in-built MATLAB functions. Version 3 replaces all MATLAB code and is compiled directly as C++ source as a library. Version 3 is much more efficient as there is no extra overhead of the MCR as in version 2. The following subsections describe the LOFT system architecture through the different versions.

6.1.1 System Architecture

Figure 6.1 shows the modular structure of Version 2 of the C++ and MATLAB compiled code interface. The MCR acts as a wrapper around any compiled MATLAB code and adds the additional overhead for initialization. Figure 6.2 shows refactored

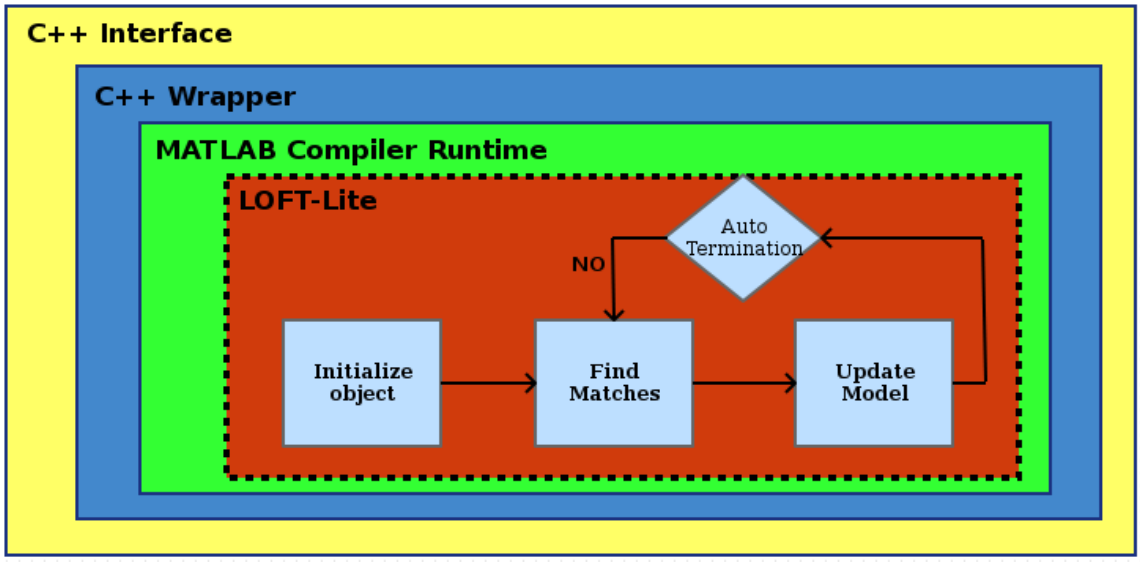


Figure 6.1: LOFT-Lite modular architecture featuring the interface to MATLAB code.

code for the next generation Version 3 codebase that was functionally modular with the focus on using potential external programs for future use.

The three main components correspond to the previously mentioned design such as a) Interface, b) Bridge (glue code) and c) LOFT core. While these modules make the LOFT system an entirely independent tracker which contains data structures to maintain state and prediction including the core feature computation, feature fusion and the appearance module, the interface can be any external program or library. The C++ based architecture, as shown in Figures 6.3, 6.4a, 6.4b, of the KWIVER tracking and event detection pipeline, is sufficiently modular and reusable that we have been able to integrate LOFT-Lite as part of several other video exploitation frameworks including AFRL E2AT [86] and ARL Advanced Video Activity Analytics program [87]

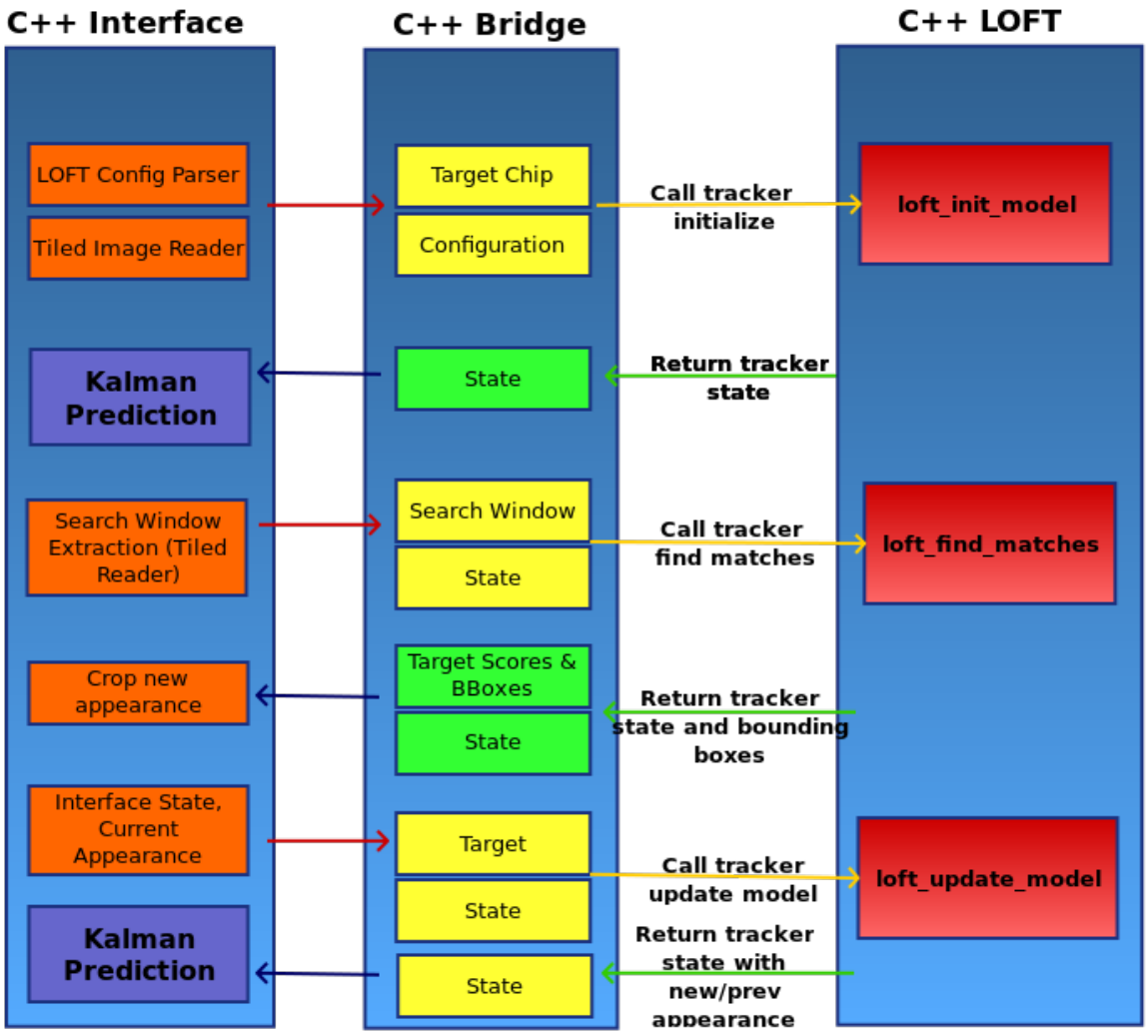


Figure 6.2: LOFT-Lite flexible framework featuring modular functionality.

This architecture as shown in Figures 6.4a, 6.4b supports the data reduction through the pipeline, starting from object detection where a significant portion of the data is excluded when it is considered as non-moving. The 3D model of the scene, a relatively newer addition, is used to distinguish between the false motion parallax from the tall scene structures and the true movers. In the urban areas such

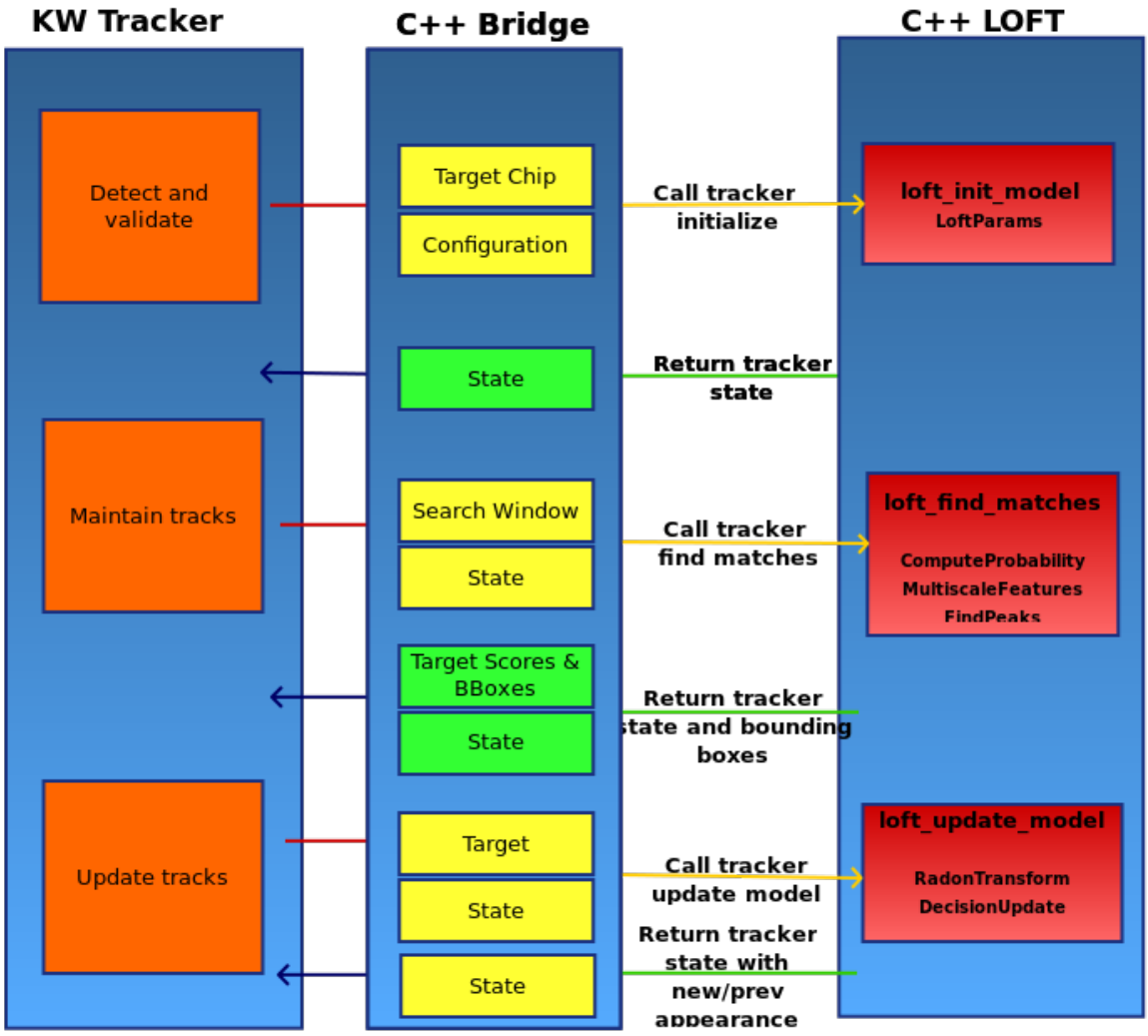
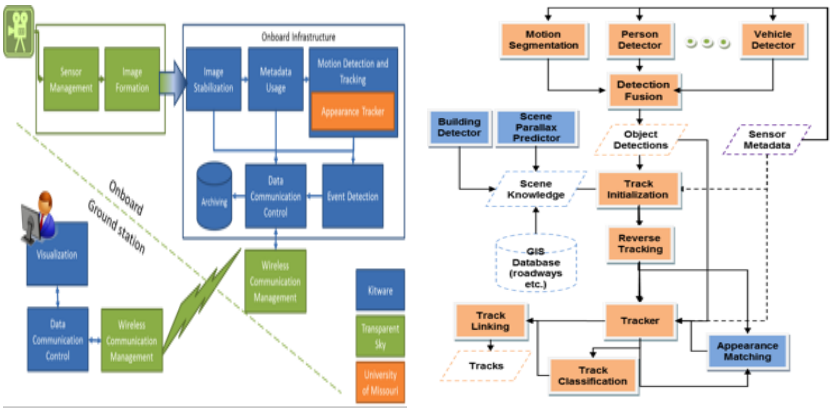


Figure 6.3: LOFT-Lite flexible framework replaces the LOFT generic interface with the KWIVER calling module. This figure was shown in Bashrat et al. [88]

a technique based on the scene model is particularly useful where the tall structures my result in severe amount of false alarm. Such a design of the tracking pipeline is scalable for processing larger amounts of data, where the onboard processing can be distributed across the GPUs and various CPU workers for the various processing tiles. Notice that the appearance matching block is a separate module, which implies



(a) Onboard processing concept used to design onboard processing modules [88]

(b) Architecture of the KWIVER tracker pipeline to integrate a future 3D model and LOFT appearance matching [88]

convenient integration of various algorithms. Our main contribution to this pipeline is called LOFT-Lite (version 3) and uses the templated CImg library (`cimg.eu`) for its internal data structures. The code that glues the KWIVER tracker with LOFT-Lite is also in C++ and functions as a translator for data structures between the native VXL types to CImg type arrays or structures. The glue code along with LOFT-Lite is compiled to a dynamic library which is then loaded at runtime by the KWIVER tracker. Currently, a fully C++ multithreaded environment has been completed and delivered for the completion of the CETE project. Figure 6.4 represents the current full flow diagram. The core functions are abstracted and named as initialize, match and update in the figure. Thus, we have formulated a generic way of interfacing any appearance based tracking algorithm in a similar way as LOFT (Figure 6.4a). We have evaluated this pipeline with several appearance based trackers and key parameter changes that are discussed in the subsequent chapter.

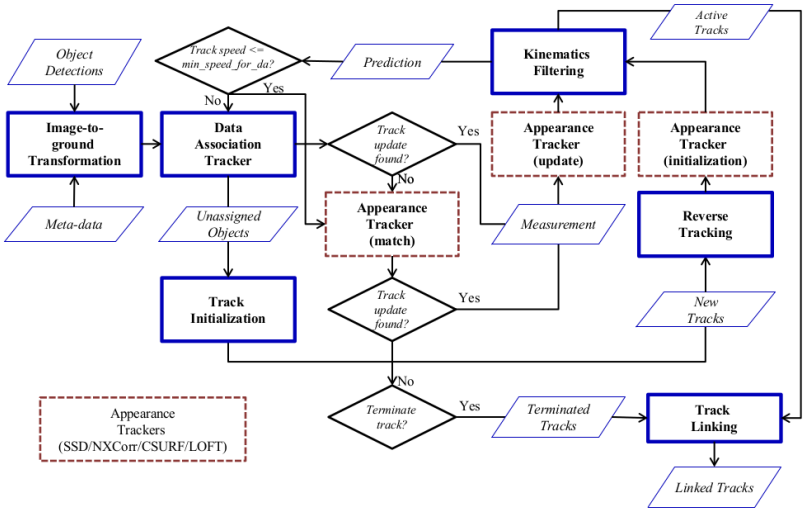
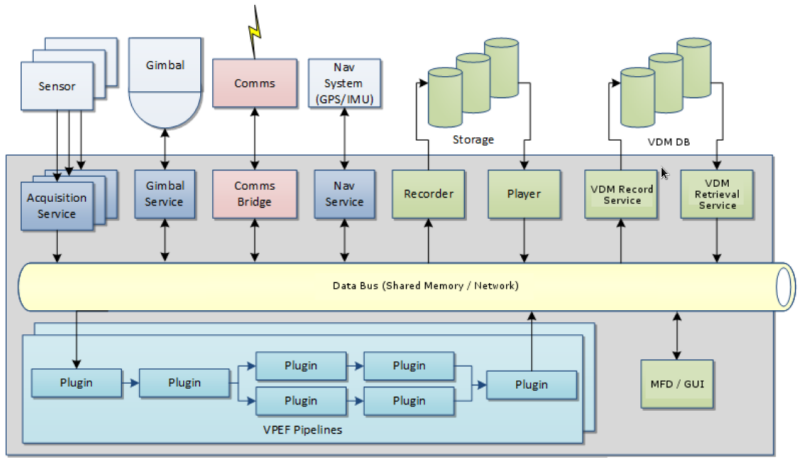


Figure 6.4: Current flow diagram of existing CETE architecture including LOFT's core functionality (initialize, match and update) [88]

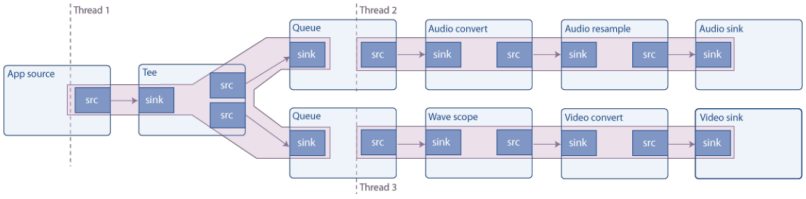
6.2 Advanced Video Activity Analytics (AVAA)

The AVAA system was designed with an objective for full Full Motion Video (FMV) exploitation capability as part of a program by the Army Research Lab [89]. The ultimate goal is to reduce the analyst's workload and to enable faster and accurate large scale video analysis [87]. Lower level algorithms will then provide feedback to abstracted high levels of correlation of data across the enterprise by automatically analyzing, annotating and processing large amounts of video streams.

Video Processing and Exploitation Framework (VPEF) [90] is an agile and flexible framework that allows for fast integration of computer vision algorithms. VPEF is based on a GStreamer framework that allows for plug-ins to be created and deployed by an analyst. It allows for maximum flexibility while retaining robustness of an end to end customized pipeline. VPEF achieves this by compartmentalizing computer vision



(a) Overview of VPEF pipeline interacting with components of data acquisition, storage, retrieval and processing [90].



(b) VPEF video streaming pipeline showing source and sinks for standardized input and output [90]. Standard UNIX tools like the use of tee to split the input and output streams derived its design from the functionality of GStreamer.

algorithms with standardized inputs and outputs. The design principles for LOFT and VPEF are similar with the major difference being that VPEF is a more generic and higher level controller while LOFT is one of the components and an available option within this framework. Figures 6.5a and 6.5b show the overall VPEF plug-in based video streaming pipeline and the internal data handling for the plug-in layer. Completely asynchronous way of handling data IO minimizes risk of failure of the entire pipeline by compartmentalizing each computer vision algorithm. LOFT has been integrated as a plug-in within the overall VPEF architecture and is available

as a plug-in that can perform appearance based tracking. The overall flexibility and state independent processing that is part of the modular implementation of LOFT assisted us in integrating with the VPEF architecture in a matter of hours.

Chapter 7

Experimental Results

The experimental methodology benchmarks the LOFT tracking algorithm as described in Chapter 5. While many of the state of the art algorithms fail to perform consistently across these two different set of images, LOFT performs well on an average with robust tracking results. LOFT as a general framework always outperforms standard baseline algorithms without tweaking parameters. The chapter is organized in three major sections. The first section describes our experimental methodology on the CLIF dataset along with detailed results compared with standard trackers in literature. In the second section, we have demonstrated with experiments the impact and usefulness of the appearance modeling and orientation estimation. The final section describes the results of the multi-target tracking platform after integration within the Kitware framework for the CETE project as described in Chapter 6 section 6.1.

7.1 CLIF : Experimental Results

LOFT was evaluated using the Columbus Large Image Format (CLIF) [78] WAMI dataset which has a number of challenging conditions such as shadows, occlusions, turning vehicles, low contrast and fast vehicle motion. We used the same vehicles selected in [9] which have a total of 455 ground-truth locations of which more than 22% are occluded locations. The short track lengths combined with a high degree of occlusions makes the tracking task especially challenging. Several examples of the difficulties in vehicle tracking in CLIF are illustrated in Figure 7.1. Figure 7.2 shows that half the sequences in this sample set of tracks have a significant amount of occluded regions and Table 7.1 summarizes the challenges in each sequence. We used several FMV sequences which have been used to benchmark a number of published tracking algorithms in the literature. These sequences include: 'girl', 'david', 'faceocc', 'faceocc2' [19] and allow comparison of LOFT against a number of existing tracker results for which source code may not be available.

7.1.1 Registration and Ground-Truth for CLIF WAMI

In our tests we used the same homographies as in [9] that were estimated using SIFT (Scale Invariant Feature Transform [91]) with RANSAC to map each frame in a sequence to the first base frame. Several other approaches have been used to register CLIF imagery including Lucas-Kanade, and correlation-based [92], or can be adapted for WAMI [93, 94]. Using these homographies we registered consecutive frames to the first frame in each sequence. The homographies when applied to the ground-truth

bounding boxes can produce inaccurate quadrilaterals since these transformations are on a global frame level. All quadrilaterals were automatically replaced with axis aligned boxes and visually inspected to manually replace any incorrect bounding quadrilaterals, on registered frames, with accurate axis aligned boxes using KOLAM [3, 4, 95] or MIT Layer Annotation Tool [96].

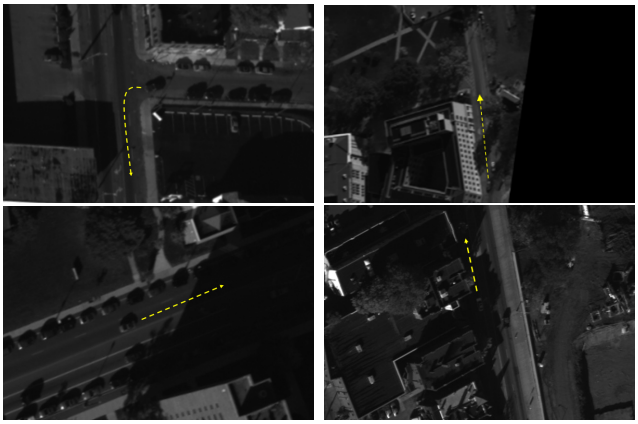


Figure 7.1: Example of challenging conditions: Target appearance changes during turning (C4-1-0), low contrast and shadows (C3-3-4), shadow occlusion (C0-3-0) and combined building and shadow occlusion (C2-4-1) [97].

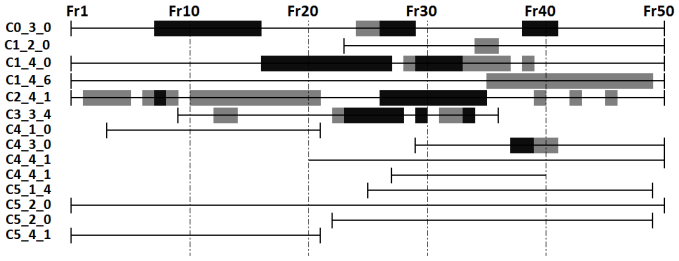


Figure 7.2: Distribution of occluded frames in the 14 CLIF seq. Black: fully occluded, Gray: partially occluded. Target is occluded in 22.4% of the frames [42].

Seq. No	Challenges	Track Length	Target Size [pixel]	Occ.Fr
C0_3_0	Occlusion	50	17x25	17
C1_2_0	Occlusion	27	21x15	2
C1_4_0	Occlusion	50	21x17	21
C1_4_6	Occlusion	50	25x25	15
C2_4_1	Occlusion	50	25x17	32
C3_3_4	Occlusion	27	27x17	12
C4_1_0	Turning car	18	15x25	-
C4_3_0	Occlusion	20	21x17	3
C4_4_1	Low contrast	30	17x21	-
C4_4_4	-	13	17x25	-
C5_1_4	Fast target motion	23	27x11	-
C5_2_0	Fast target motion	49	21x15	-
C5_3_7	-	27	27x47	-
C5_4_1	Low Contrast	21	27x19	-
Total		455		102

Table 7.1: Characteristics of the 14 CLIF sequences summarized from [9] showing track length, vehicle target size and number of occluded frames. Image frames are 2008×1336 pixels.

7.1.2 Quantitative Comparison

We used several measures of performance to quantitatively evaluate the trackers. The first one is the Missing Frame Rate (MFR) which is the percentage of number of *missing* frames to the total number of ground-truth frames,

$$\text{MFR} = \frac{\#\ missing\ frames}{\#\ total\ \ GT\ \ frames} \tag{7.1}$$

A frame is labeled as *missing*, if the predicted/estimated object location with associated bounding box overlaps with the ground-truth by less than 1% or there is no bounding box at all, for example due to early track termination. The one percent overlap threshold is the correct one that was actually used in the CLIF experiments reported in Ling *et al.* [9] (not 50%). We used bounding boxes of roughly the same size as the target at the predicted location; note that MFR does not explicitly penalize the use of large bounding boxes.

Two commonly used criteria are precision and recall scores for the tracker predicted/estimated (single) target locations [98]. Precision (related to track purity) is defined as the ratio of the number of correctly tracked frames, $|TP|$, to total number of tracked frames or track length,

$$\text{Precision} = \frac{\#\ correct\ frames}{\#\ tracked\ frames} = \frac{|TP|}{|TP| + |FP|} \tag{7.2}$$

where number of correct frames are those in which target locations are within a set threshold distance from the ground-truth (*i.e.* 20 pixel radius ribbon). Recall (related to target purity) is the ratio of number of correctly tracked frames to number of ground-truth frames for the target defined as,

$$\text{Recall} = \frac{\#\ correct\ frames}{\#\ GT\ track\ frames} = \frac{|TP|}{|TP| + |FN|} \approx 1 - MFR. \tag{7.3}$$

The equality is approximate since MFR uses a bounding box overlap criteria whereas precision and recall use a distance from ground-truth centroid criteria. The final performance metric used for evaluating tracking performance is the tracking position

CLIF Seq.	MIL [20]	MS [99]	CPF [100]	HPF [101]	L1-BPR [9]	NN [19]	PN [21]	LOFT
C0_3_0	0.860	0.980	0.940	0.980	**0.760**	0.920	0.940	**0.740**
C1_2_0	0.852	0.963	0.9636	0.963	0.630	0.962	0.962	**0.000**
C1_4_0	0.680	0.780	0.740	0.760	**0.620**	1.000	0.700	0.720
C1_4_6	**0.360**	0.940	0.800	0.880	**0.360**	0.980	0.560	0.580
C2_4_1	0.900	0.980	0.980	0.980	0.920	0.980	0.980	**0.877**
C3_3_4	0.963	0.963	0.963	0.963	0.704	0.962	0.962	**0.370**
C4_1_0	**0.389**	0.889	0.833	0.889	**0.389**	0.888	0.944	0.611
C4_3_0	**0.650**	0.950	0.950	0.800	0.750	0.947	—	**0.005**
C4_4_1	0.533	0.967	0.900	0.900	**0.033**	0.931	0.758	**0.000**
C4_4_4	**0.000**	0.923	0.385	0.307	**0.000**	0.076	0.923	**0.000**
C5_1_4	0.667	0.958	0.875	0.833	0.667	0.958	0.958	**0.000**
C5_2_0	0.918	0.979	0.959	0.979	0.979	0.979	—	**0.062**
C5_3_7	**0.000**	0.963	0.148	**0.00**0	**0.000**	0.8516	0.259	**0.000**
C5_4_1	**0.000**	0.952	0.810	0.905	0.958	0.523	0.809	**0.000**
Mean	0.555	0.942	0.803	0.796	0.555	0.854	0.813	**0.287**
OverAll	0.627	0.940	0.833	0.837	0.611	0.909	0.680	**0.333**

Table 7.2: Missing frame rate (MFR) performance (lower the better) on CLIF WAMI data. Results for Multiple Instance Learning Tracker (MIL), Mean Shift tracker (MS), Covariance Based Particle Filter (CPF) tracker, Histogram-based Particle Filter (HPF) tracker and ℓ_1-Bounded Particle Resampling (L1-BPR or Sparse) tracker are from Ling *et al.* [9]. Mean indicates average of sequence MFRs (shorter tracks have higher influence) while OverAll is an ensemble average as in [9].

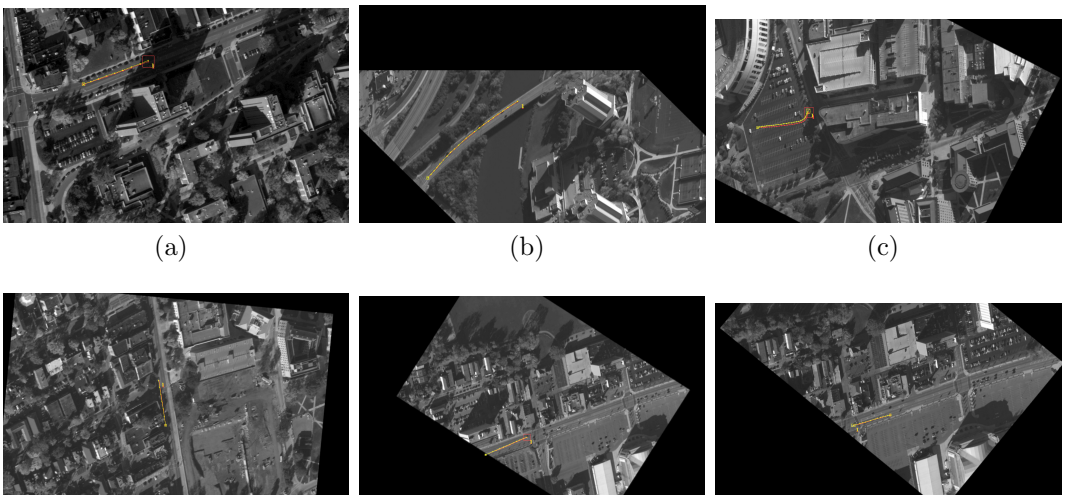
errors defined as the distance between the predicted object position and the ground-truth centroid. Track completeness, fragmentation, mean track length, id switches and other measures of multi-target tracking performance are necessary for a more thorough evaluation of tracking performance [11].

LOFT performance was compared to several state-of-the-art trackers. Some of the LOFT modules (see Figure 3.1) were turned off for the experiments including binary classifier, background subtraction, and MHT in order to focus on evaluating the appearance update performance. The tracking methods we ran using author provided source code in the experiments are Nearest-Neighbor (NN) Tracker [19], L1-BPR Sparse Tracker [24], Multiple Instance Learning (MIL) Tracker [20], and P-N Tracker [21]. We did some limited parameter tuning for optimizing each tracker

Method	Precision	Recall
L1-BPR [24]	0.185	0.185
MILTrack [20]	0.271	<u>0.271</u>
P-N [21]	<u>0.373</u>	0.172
NN [19]	0.088	0.082
LOFT	**0.603**	**0.405**

Table 7.3: OverAll Precision - Recall scores across 14 CLIF sequences. Second best performance underlined [42].

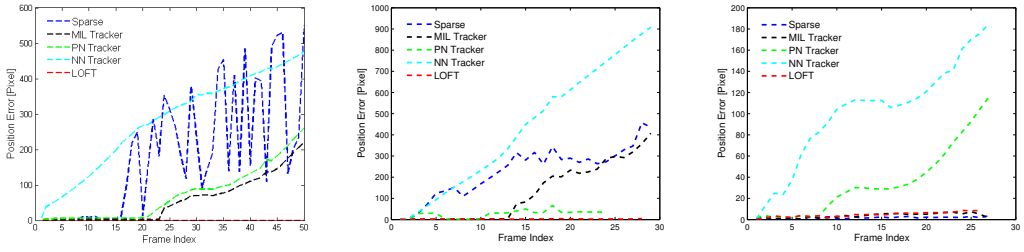
for both CLIF and FMV separately. Table 7.2 summarizes the MFR scores of these five trackers on CLIF data. Table 7.3 shows the overall precision-recall scores for the 14 sequences in the CLIF dataset. Figure 7.4 shows position errors three sample CLIF sequences where LOFT does particularly well. These comparisons show that our LOFT tracker outperforms all other trackers on this CLIF dataset. According to the MFR scores, MILTrack and L1-BPR Sparse trackers produced comparative results for some of the sequences, however, the lack of a termination module causes their precision scores to drop significantly in Table 7.3. The P-N tracker has very good performance on FMV, but the search method involves scanning the entire image and thus testing on WAMI posed severe memory constraints. P-N tracker has the second highest precision on the CLIF data. The NN tracker had the worst results on CLIF WAMI likely due to the need to tune the SIFT features. Figure 7.3 shows some visual trajectories of tracking results where LOFT does well. Two sequences where LOFT did not do well, are C0_3_0 which is challenging for all of the trackers, and C2_4_1 which has many nearby spatial and temporal distractors while turning or strong occlusions; see Figure 7.1 for the visual appearance of these targets and environments.



(a) (b) (c)

(d) (e) (f)

Figure 7.3: LOFT results (red tracks) for six sequences (First row: C0_3_0, C1_2_0, C1_4_6, Second row: C2_4_1, C4_4_1, and C4_4_4) showing enhanced images with ground-truth tracks in yellow. These are six of the nine sequences in which LOFT outperforms other trackers. This illustration was created using Kolam [4] and was shown in Pelapur et al. [42]



(a) (b) (c)

Figure 7.4: Position error over the entire sequence in pixels versus frame index for five of the trackers on three selected CLIF sequences [42] (C1_4_6, C4_4_1 and C5_3_7) for which LOFT has a high accuracy.

Sequence	PROST [102]	AdaBoost [103]	FragTrack [28]	L1-BPR [23]	MILTrack [20]	P-N [21]	NN-Track [19]	LOFT
Girl	19.00	43.30	26.50	67.84	31.60	28.88	<u>18.00</u>	**13.86**
David	<u>15.30</u>	51.00	46.00	63.12	15.60	**10.38**	15.60	40.6
Faceocc	<u>7.00</u>	49.00	**6.50**	20.78	18.40	13.99	10.00	10.79
Faceocc2	17.20	19.60	45.10	73.27	14.30	19.14	**12.90**	<u>13.25</u>

Table 7.4: Mean Position Errors on standard full motion Videos with Prost, Adaboost and FragTrack results from Gu *et al.* [19]. Best results and second best results are shown in bold and underlined respectively [42].

Since most published trackers are designed for standard FMV sequences, we also evaluated LOFT on several popular benchmark videos with very different scene content and characteristics compared to WAMI. Table 7.4 shows the mean distance error to ground-truth for eight published trackers including LOFT, on four standard FMV sequences across all frames of each sequence. The PROST, AdaBoost and FragTrack results are taken from Gu *et al.* [19]. Figure 7.5 shows sample frames from the tracking results of LOFT compared to GT for 'girl' and 'faceocc2' sequences. Instead of tight initial bounding boxes we used the actual GT bounding box for the appropriate start frame in each FMV sequence. Based on the mean distance errors, the LOFT system is comparable to the other trackers on these four representative FMV sequences. LOFT also produced better results than the other trackers for 'girl' and 'faceocc2' sequences.

7.2 Adaptive Appearance Evaluation on CLIF

The impact of using adaptive appearance updating on LOFT performance was assessed using the challenging CLIF dataset [78]. We used the same 14 sequences as in previous papers [9, 42] which provide more details about the targets and sequences. We point out that the original CLIF sequences contain duplicated frames in some

Figure 7.5: Tracking results [42] showing sample frames from 'girl' and 'faceocc2' sequences showing bounding boxes for ground-truth (yellow) and LOFT (red).

cameras which adversely affects Kalman filter predictions of target position. The duplicated frames is a sort of saccadic masking effect wherein missing or corrupted image data has been replaced with temporal duplicates. This is not particular to this dataset alone but is commonly used during video capture of large sequences as it is a way of mitigating the problem of out of order sequences between metadata and the raw images. Our resulting solution of temporal image differencing to check for duplicates is a solution which can help in all datasets. LOFT tracking performance in our tests used the same modules for feature fusion and Kalman filter motion prediction (offset) as described in [2, 42, 72] without background subtraction, local registration (since the data was registered off-line), vehicle classification or MHT. We use precision and recall scores as a metric where precision measures the degree of repeatability

of a video-tracking result [98],

$$\text{Precision} = \frac{|TP|}{|TP| + |FP|}. \tag{7.4}$$

Recall is the ratio of the correctly tracked frames to ground-truth frames,

$$\text{Recall} = \frac{|TP|}{|TP| + |FN|}. \tag{7.5}$$

For a frame to be marked as correctly tracked for precision-recall metrics, the center was required to be within 20 pixels of the ground-truth center. We also use the Missing Frame Rate (MFR) metric [9], with a 1% overlap between the tracking bounding box and the ground-truth box as the match criteria; note that MFR has no explicit penalty for the size of the box. Since the two match criteria are different, $MFR \approx 1 - Recall$. Mean distance is defined as the distance between the last tracked point in a track generated by a tracker to the corresponding point in ground-truth. Note that this metric does not penalize the tracker for the remaining ground-truth. Table 7.5 shows the tracking results with and without the update scheme. It can be seen that the tracker using appearance updating consistently outperforms the one without the update scheme, with average precision and recall being 3% and 11% higher respectively. The average MFR is about 12% lower with orientation estimation based appearance modeling. Table 7.6 shows the MFR as compared with other standard trackers in literature. LOFT consistently outperforms the trackers such as Multiple Instance Learning [20], L1-BPR [24], Tracking-Learning-Detection [30] and Struck [31]. Figure 7.7 shows a summary of all results. The film-strip view was

generated by sampling frames of a given object, estimating the orientation and the frame at which the tracker terminated. It should be noted that the images have been contrast enhanced for ease of viewing. For better visualization, Figure 7.6 shows the orientation on the first frame which is represented as a tiled set of images showing all 14 vehicles in the CLIF sequence along with their variance plots with the estimated angle marked in red.

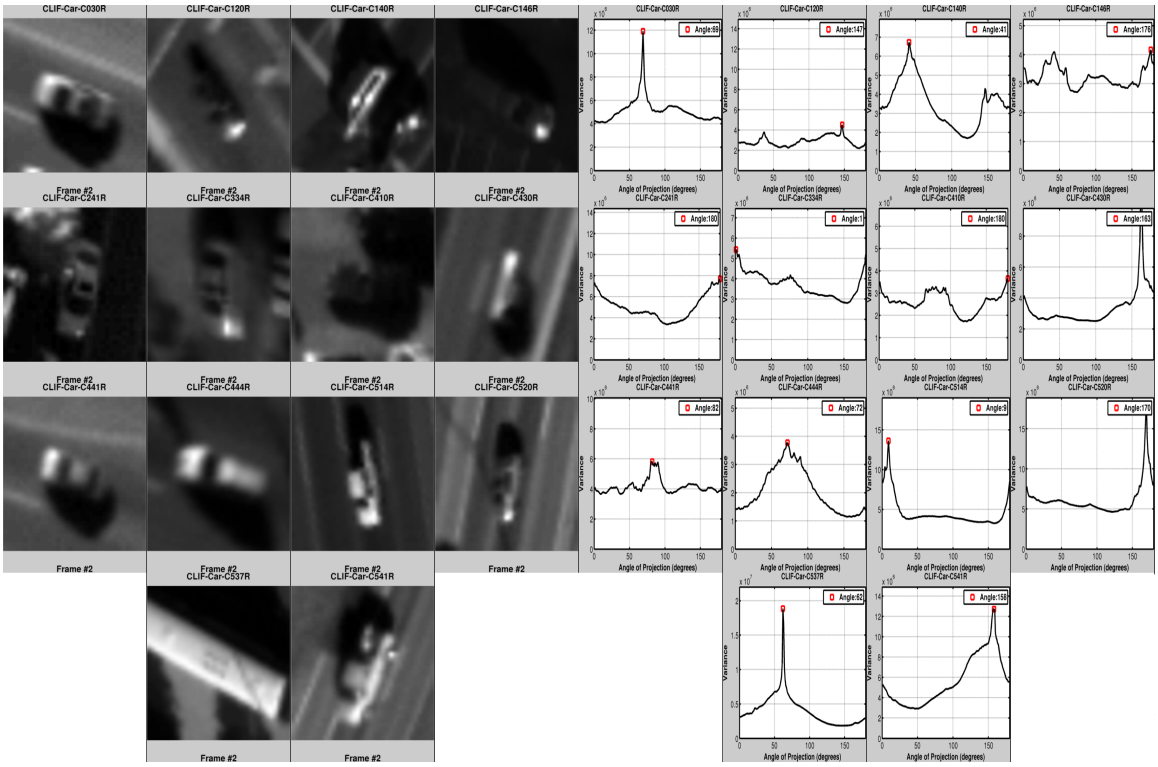


Figure 7.6: Figure shows all 14 CLIF objects in grid format with real templates (contrast enhanced) and their corresponding variance plots with labeled maximum valued peak

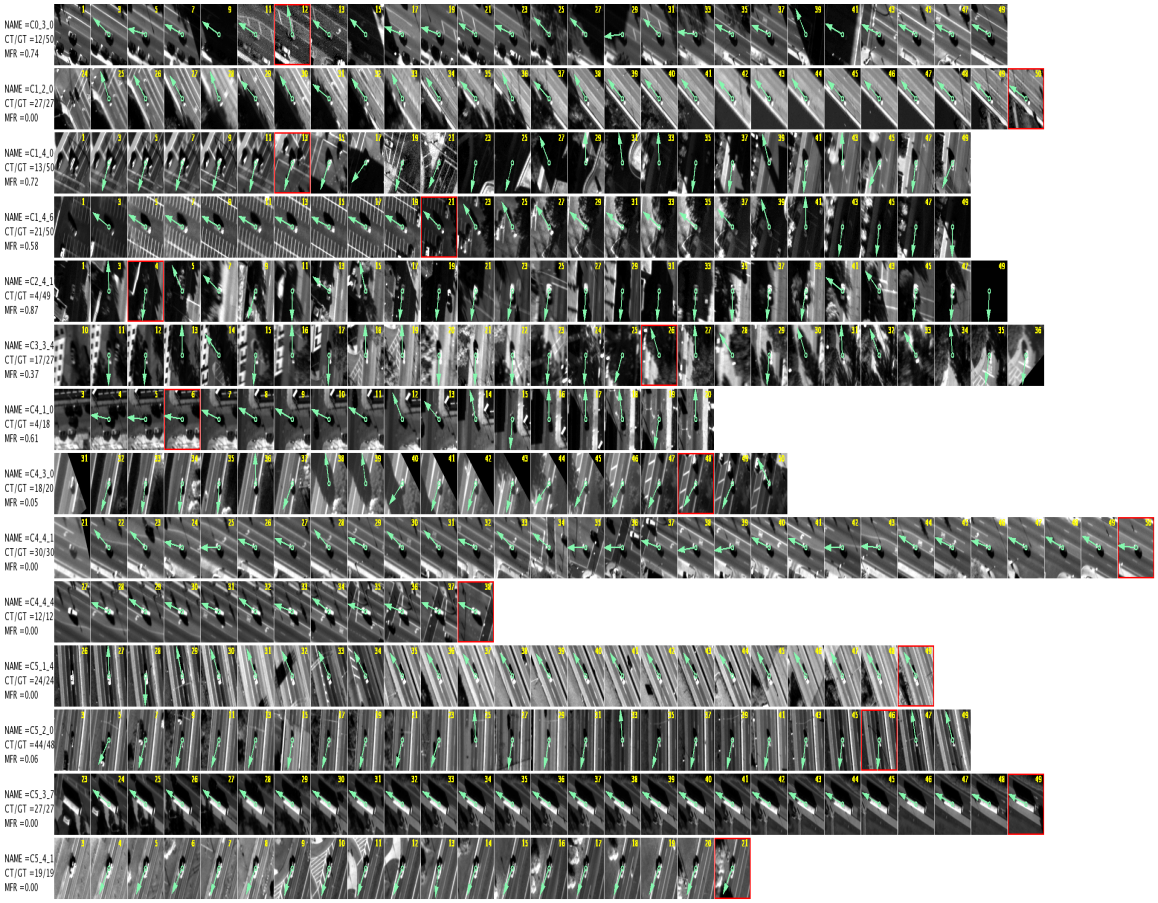


Figure 7.7: Figure showing detailed results of all 14 CLIF objects where estimated orientation is denoted by arrows, the sampled frames are numbered in yellow and the last correctly tracked frame (as compared with groundtruth) is marked with red borders.

Image	GT	Track Length		Correctly Tracked		False Alarms		Precision		Recall		Mean Distance		MFR	
Sequence		w/o update	w update	w/o update	w update	w/o update	w update	w/o update	w update	w/o update	w update	w/o update	w update	w/o update	w update
C0_3_0	50	13	14	13	12	0	2	1.000	0.857	0.260	0.240	3.916	12.72	0.740	0.740
C1_2_0	27	27	27	20	26	7	1	0.741	0.963	0.747	0.963	45.94	3.735	0.222	0.000
C1_4_0	50	12	22	12	13	0	9	1.000	0.591	0.240	0.260	4.607	68.87	0.760	0.720
C1_4_6	50	41	29	21	21	20	8	0.512	0.724	0.420	0.420	270.9	177.2	0.580	0.580
C2_4_1	49	8	12	3	4	5	8	0.375	0.333	0.061	0.082	33.08	35.80	0.918	0.877
C3_3_4	27	21	17	14	17	7	0	0.667	1.000	0.518	0.630	42.37	4.454	0.481	0.370
C4_1_0	18	8	16	4	4	4	12	0.500	0.250	0.222	0.222	24.12	93.98	0.611	0.611
C4_3_0	20	18	20	15	18	3	2	0.833	0.900	0.750	0.900	17.59	5.930	0.250	0.050
C4_4_1	30	30	30	29	30	1	0	0.967	1.000	0.967	1.000	2.389	1.857	0.000	0.000
C4_4_4	12	12	12	12	12	0	0	1.000	1.000	1.000	1.000	0.966	1.333	0.000	0.000
C5_1_4	24	14	24	8	24	6	0	0.571	1.000	0.333	1.000	50.47	2.444	0.625	0.000
C5_2_0	48	30	45	27	44	3	1	0.900	0.978	0.562	0.917	4.843	4.136	0.396	0.062
C5_3_7	27	27	27	27	27	0	0	1.000	1.000	1.000	1.000	4.471	5.286	0.000	0.000
C5_4_1	19	19	19	19	19	0	0	1.000	1.000	1.000	1.000	1.982	2.534	0.000	0.000
Average		20	22	16	19	4	3	0.790	0.828	0.577	0.688	36.26	30.02	0.399	0.287
Overall								0.800	0.863	0.497	0.601				

Table 7.5: LOFT performance on 14 CLIF sequences with and without appearance updates showing Correctly Tracked frames (higher is better), False Alarms, Precision, Recall, Mean Distance error (smaller is better), and Missing Frame Rate (MFR). Text in bold highlights the better result. Table 7.2 published in [42] had a mistake in the Missing Frame Rate (MFR) computation. Since then the computation was corrected, more accurate groundtruth was generated and several improvements were made to LOFT causing the change in the results.

Sequence	MIL [20]	L1-BPR [24]	TLD [30]	STR [31]	LOFT-w/o-update	LOFT-w-update
C0_3_0	0.860	0.760	0.920	0.980	**0.740**	**0.740**
C1_2_0	0.851	0.629	0.962	0.629	0.222	**0.000**
C1_4_0	0.680	**0.620**	0.700	1.000	0.760	0.720
C1_4_6	**0.360**	**0.360**	0.540	0.260	0.580	0.580
C2_4_1	0.900	0.920	0.979	0.979	0.918	**0.877**
C3_3_4	0.963	0.703	0.962	0.962	0.418	**0.370**
C4_1_0	**0.388**	**0.388**	0.944	0.888	0.611	0.611
C4_3_0	0.650	0.750	0.950	0.850	0.250	**0.005**
C4_4_1	0.533	0.033	1.000	0.966	**0.000**	**0.000**
C4_4_4	**0.000**	**0.000**	0.916	**0.000**	**0.000**	**0.000**
C5_1_4	0.666	0.666	DNR	0.958	0.625	**0.000**
C5_2_0	0.918	0.979	0.979	0.958	0.396	**0.062**
C5_3_7	**0.000**	**0.000**	0.962	0.000	**0.000**	**0.000**
C5_4_1	**0.000**	0.958	0.894	0.894	**0.000**	**0.000**
Average	0.555	0.555	0.901	0.737	0.399	**0.287**

Table 7.6: Table showing results of state of the art trackers such as Multiple Instance Learning [20], L1-BPR [24], Tracking-Learning-Detection [30] and Struck [31], in literature on WAMI-CLIF evaluated against LOFT with and without update

7.3 Multitarget Tracking Results

	LOFT-only	CSURF-only	Template-only	Motion-only	LOFT-Hybrid	CSURF-Hybrid
Detection-Pd:	0.37	0.41	0.40	0.37	**0.44**	**0.44**
Detection-FA:	*213*	429	1144	**35**	488	397
Track-Pd:	**0.95***	**0.95***	**0.95***	**0.95***	**0.95***	**0.95***
Track-FA:	*3*	*3*	*3*	**2**	**2**	**2**
Avg. track purity:	0.95	*0.97*	0.95	**0.99**	0.93	*0.97*
Avg. target continuity:	4.45	2.85	4.7	2.7	**2.1**	2.5
Avg. target purity:	0.24	0.38	0.27	0.37	**0.49**	*0.43*

Table 7.7: ARGUS-IS tracking results through the proposed tracking system. Best results in **Bold**, second best results in *Italics*. The proposed fusion of LOFT and CSURF with motion produces the best results. * missed one target due to a very short truth track. These results were published in Bashrat et al. [88]

The proposed tracking framework is implemented in C++ with LOFT and CSURF initially implemented in Matlab, and integrated through dynamic library interface. LOFT implementation was later revised as a C++ based library and loaded through the appearance tracker interface.

LOFT and CSURF algorithms were evaluated on the ARGUS-IS dataset, a sample frame with detailed results is shown in Figure 7.8. This dataset included 1000 frames @ 3Hz, with 19 manually annotated truth tracks for analyzing vehicle tracking. The vehicles were annotated when they started moving and continued the annotation when they stop momentarily or get parked. ARGUS-IS platform is able to capture Wide-Area Motion Imagery (WAMI) over 40 square kilometers with a Ground Space Distance (GSD) of 15 cm at video rates of greater than 12 Hz [104]. The frame-rate of the dataset used here was limited to 3Hz, which makes the tracking more difficult than the full frame-rate. In the past, ARGUS-IS data has been successfully processed for surveillance and interpreting object behavior to recognize functional elements of the scene [105].

Figure 7.8: ARGUS-IS data showing LOFT tracks on frame #223 (full scene as in Left-Bottom (LB)-purple). Three zoomed up insets are marked with colored boxes (Right-Middle (RM)-yellow, Right-Bottom (RB)-light red, Top (T)-cyan) and each of them show the area that was zoomed into in the overview image (LB-purple). These tracking results were generated using **LOFT-Hybrid** (*min_speed_for_da*= 3) configuration, same as Table 7.8. Notice that in the (T) inset the objects that are being tracked are those that were moving in the previous time steps and are now stationary while two other moving objects (RM) & (RB) are in motion showing the adaptive integration of appearance and motion model. This illustration was shown in Bashrat et al. [88]

The analysis of the track quality was performed by the commonly used metrics based on probability of detection (Pd) and false alarm (FA). Our software for scoring is available as KWANT open source tool under the KWIVER toolkit [106]. Metrics used for scoring tracks are defined below along with the numerical ranges: *Track-Pd* [0,1], ideal value = 1, number of computed tracks overlapping with true tracks / Number of true tracks; *Track-FA* [0,∞), ideal value = 0, number of computed tracks not overlapping with true tracks; *Detection-Pd* [0,1], ideal value = 1, number of detections in computed tracks overlapping with true tracks / Number of detections in true tracks; *Detection-FA* [0,∞), ideal value = 0, number of detections in computed tracks not overlapping with true tracks; *Target Continuity* [1,∞), ideal value = 1, number of tracks initialized on a given target; *Target Purity* [0,1], ideal value = 1, percentage of associations with the predominant track utilizing the given target over the life of the target. Continuity and purity are also defined for the tracks similar to that for the targets.

Table 7.7 presents our main result as the measure of track quality under various algorithmic configurations: using appearance only information (LOFT-only, CSURF-only & Template-only); no appearance information (Motion-only); and fusion of appearance based updates with the motion detections to update tracks (LOFT-Hybrid and CSURF-Hybrid). The first three configurations (columns) show the quality of the tracks when the track update was performed only based on either LOFT, CSURF or template matching (SSD) appearance trackers. Next column shows the performance when only motion detections are used through data association to update tracks. This is following by the hybrid of motion with LOFT or CSURF appearance track-

ers based on *min_speed_for_da* value of 3 m/s. The least amount of *Detection-FA* is produced by Motion-only, which is consistent with a very conservative tracker with lowest *Detection-Pd* and is not even expected to track moving cars through stops. Overall LOFT-Hybrid seems to be producing the best results based on the majority of the metrics showing this as the best configuration. CSURF-Hybrid is probably the second best configuration, depending on the application the tracks are used in. This shows that proposed fusion of appearance and motion detections is feasible and suitable for the data analyzed.

Next, we analyze the impact of the *min_speed_for_da* parameter on the fusion between LOFT and motion detections. As shown in Figure 6.4, the system determines whether motion detections should be considered for track update before appearance tracker at low target speeds when the motion signature might be unreliable. The impact of this parameter was studied in an experiment that involved sweeping a range of values to analyze the impact on the track quality. As can be seen in Table 7.8, 0 m/s produces best results across most of the metrics; in this case motion detection is preferred over LOFT and LOFT will be only used when the car is stationary. Note that the configurations that are directly comparable are column #6 (LOFT-Hybrid) in Table 7.7 and column #3 (3 m/s) in Table 7.8. The significant difference in Detection-Pd between Table 7.7 and Table 7.8 is mainly due to rolling changes made in the overall tracking pipeline. We attribute this difference to the changes such in the C++ implementation and improvement of LOFT, ongoing improvements to the overall software pipeline, and changes made to the scoring program. 3 m/s configuration, a close second best here, generalizes very well in our experience on other

Min. target speed motion det.(*min_speed_for_da*)	**0 m/s**	**3 m/s**	**5 m/s**	**10 m/s**
Detection-Pd:	**0.72**	0.68	0.43	0.34
Detection-FA:	**341**	398	537	411
Track-Pd:	***0.95***	***0.95***	***0.95***	***0.95***
Track-FA:	2	2	2	**1**
Avg. track purity:	***0.96***	***0.96***	0.95	***0.96***
Avg. target continuity:	**2.35**	2.40	2.75	6.25
Avg. target purity:	***0.52***	***0.52***	0.44	0.24

Table 7.8: Analysis of various speed thresholds to fuse motion detections and LOFT, as shown in Figure 6.4, to update tracks. In ARGUS-IS data, the best configuration to use LOFT seems to be on the stationary cars. These results were published in Bashrat et al. [88]

datasets. These datasets with lower resolution, and lower contrast gray-scale imagery seem to generally have inferior motion detection quality at lower target speeds, where using LOFT and CSURF improves results [22, 77].

In this chapter we have shown that LOFT with appearance modeling outperforms many of the standard trackers in literature. It can also track on varied datasets and helps in multitarget tracking as an appearance module. We believe that these results showcase the robustness of LOFT as a flexible algorithm.

Chapter 8

Scalable Tracking Using Visual Cloud Computing

Cloud computing is a powerful technique to leverage the use of large off-site data processing [107]. As in the case of wide-area tracking the amount of input data is large but the output data is restricted to very sparse information. In case of fragmented processing where individual components within the tracker core functionality can be spread across various nodes for distributed processing, the output data could include the tracker state. The tracker state as shown in Chapter 6 Figure 6.3, in case of LOFT-Lite, is a structure that contains essential information and is the only data structure that is required to continue tracking. The core functions of LOFT-Lite are stateless and is therefore an ideal candidate for distributed processing. In this chapter we describe an incident-supporting visual cloud computing solution by defining a collection, computation and consumption (3C) architecture supporting fog computing at the network-edge close to the collection/consumption sites, which is coupled with

cloud offloading to a core computation, utilizing software-defined networking (SDN). The evaluation mainly consists of looking at how SDN can help in the case of large, medium and small scale images, in terms of resolution or number of pixels, using LOFT-Lite as a case study. The main problem that can be tackled using such a system is when the collection of large scale imagery is in a geographically different location to where LOFT-Lite is hosted. Using the work done in integrating LOFT-Lite within a cloud computing architecture is important in providing object tracking software as a service. The experimental testbed also demonstrates the use of SDN for on-demand compute offload with congestion-avoiding traffic steering to enhance remote user Quality of Experience (QoE). The work described in this chapter focuses on our collaborative work with a team comprised of faculty members and graduate students [107].

8.1 Visual Cloud Computing

Computer vision commonly deals with the processing of large data sets, and a typical system in this field usually comprises of several data processing stages such as: (a) acquisition, (b) pre-processing, (c) analysis, and (d) post-processing. Data requirements change depending on the application in question, and the acquisition step itself usually requires an enormous amount of storage apart from the bandwidth requirements for processing. Separating storage and bandwidth requirements could greatly benefit overall processing time required for a data set. However, the processing time of applications can also have some restrictions based on the location at which they

are hosted. In most cases, it is scalable to have data sent over to a cloud-hosted application host have it processed and have the analysis results sent back to the origin. Large-scale visualization and analysis such as NVIDIA's Grid Computing [108] have gained traction in the consumer market. Our work fosters the trend where multimedia cloud computing discussed in [109–112] can provide high flexibility and mobility to the end user. Demonstrations of similar systems exist in literature and have been shown to work in an environment where hardware resources at data origin are limited [113, 114].

8.2 Disaster Management

During a disaster, the standard requirements such as storage, networks and software libraries may not be available. Providing an off-site service for processing algorithms including a reliable streaming platform for images when the available network paths are intermittently available, damaged or unavailable within the geographic location of the incident scene is important to analyze data. Emerging techniques in the field of mobile visual cloud computing are well suited for scalable processing of media-rich visual data [115]. Private cloud 'fogs', as well as overlay network paths that are dynamically constructed using software-defined networking (SDN) [116, 117] rely on non-traditional network protocols such as OpenFlow [118]. These can be valuable in the case of damaged or congested network infrastructure within the geographical area of incidents. Fog computing extends cloud computing closer to the network-edge locations of users and data sources. Coupled with SDN, fog computing at the edge

can rapidly compute and organize small instance processes locally and move relevant data from the incident geographical location to core cloud platforms such as Amazon Web Services or NSF Global Environment for Network Innovations (GENI) [119] for on-demand processing. Moreover, the overlay network paths can also be useful for moving cloud-processed data closer to the locations of first responders for content caching at fogs to enable low-latency access via thin-client desktops. Such on-demand computation and integration of thin-clients for visualization can enable large data processing within the cloud and deliver high user Quality of Experience (QoE).

8.3 Fog Computing

Many distributed computing applications benefit by leveraging fog computing in terms of reduced service latency and operational efficiency. For instance, Jiang et al. [120] benefited from the paradigm of fog computing in their efforts to optimize web page performance by caching information at various fog nodes, versus using the traditional content-delivery network platforms. Fog resource management solutions are proposed in [121] to handle resource allocation and pricing based on user application profiles. Interestingly, SDN has been leveraged in context of fog computing recently by Stojmenovic et al. [122], where they studied benefits of fog computing in application scenarios such as Smart Grid, and smart traffic lights in vehicular networks. Another notable recent work that leveraged SDN integrated with fog computing is [123], where benefits were shown in the context of vehicular adhoc network cases to enhance resources utilization and decrease service latency. Our work lever-

ages fog computing paradigm in the context of mobile cloud configuration with SDN for disaster incident response scenarios, and shows benefits when handling media-rich and data-intensive visual computing applications for situational awareness of first responders.

8.4 SDN Management

Several studies have been done in prior works on SDN and cloud computing for overlay network provisioning. Authors in [124] propose a new method to manage Quality of Service (QoS) requirements of applications over SDN-enabled networks based on multi-path routing. Their multi-path routing assumes intermediate hosts to run agents that support their approach to allocate resources effectively by increasing the search space for the idle resources. In the context of multimedia delivery over large-scale SDN paths, Egilmez et al. [125] proposed a distributed OpenFlow-based QoS architecture involving co-ordination of multiple controllers. Another related work can be found in [126], where an adaptive routing approach is described to handle QoS requirements of video streaming utilizing SDN. They divide the QoS flows into two levels (base layer packets and enhancement layer packets), and provide highest priority to the base layer to reroute via feasible path in case of the congestion in the shortest path. Lastly, another exemplar related work on using SDN for video flow handling can be seen in [127], where a QoS Controller (Q-Ctrl) system is used to control and allocate bandwidth for the virtual machines supporting video streaming in a cloud infrastructure. This work builds on earlier methodology Calyam et al. [128] on wide-

area experimental testbeds such as GENI [119] and extends it for the WAMI data processing context with OpenFlow based SDN controller implementations for path computation and flow steering to improve user QoE.

8.5 Collection, Computation and Consumption (3C) model

A collection, computation and consumption (3C) architecture is shown in Figure 8.1. Our design of the architecture assumes incident videos or images are collected and pre-processed at a fog near the disaster scene and are transferred utilizing SDN to cloud servers where visual analytics such as 3D geometry, object recognition and tracking can be performed. The 3D visual environment, object and tracking results are subsequently transferred from the core cloud servers to a fog near first responder mobile devices or thin-client desktops for crucial visual data consumption. Based on this 3C architecture, we proposed a novel computation placement, and SDN control algorithms designed to enable fog computing closer to the collection or consumption sites, which is coupled with cloud offloading to a public cloud. The algorithms assume the fogs are capable of handling small instance visual processing functions, and are integrated with a public cloud infrastructure for handling large instance visual processing functions by utilizing SDN.

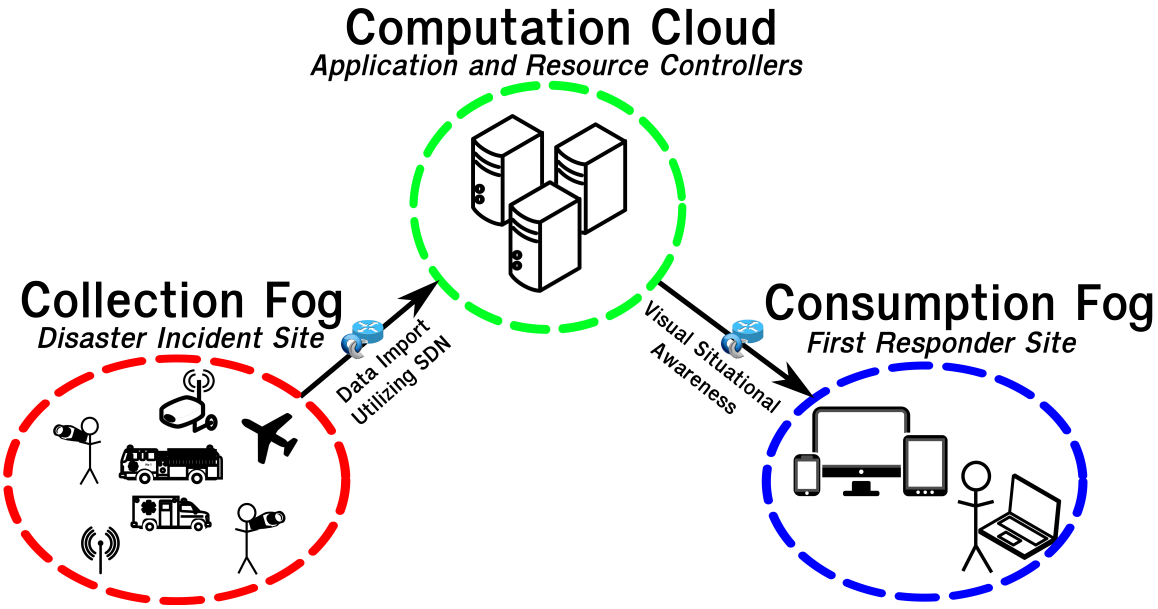


Figure 8.1: Overview of the visual collection, computation and consumption (3C) system linking the fog at the network-edge with core cloud computing utilizing SDN which is shown on the links [107].

8.6 LOFT-Lite: A Regional-Scale Application

Tracking in WAMI involves several pre-processing steps that have been tested on large-scale aerial data [94, 129] as shown in Figure 8.2. These steps can be divided into two main classes according to functionality such as:

- **Small instance processing**: Compression, storage, metadata processing, geo-projection, stabilization and tiling
- **Large instance processing**: Initialize objects of interest, detection, tracking and event analysis

Small instance processing classes mainly focus on pure pixel level information. Large instance processing classes however, focus on pixel as well as object level information.

Most of the large instance functions are dependent on the pre-processing stages in order to work effectively. As an example, most trackers need the imagery to be stabilized in order to produce the best results and hence registration becomes a key pre-processing step.

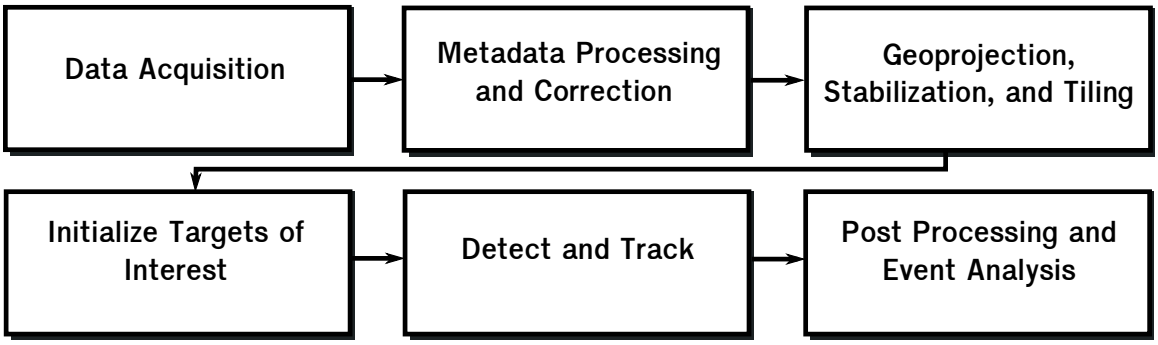


Figure 8.2: Functional block diagram showing pre-processing and post processing steps in a typical WAMI analysis pipeline [107].

LOFT-Lite as a version of LOFT [2, 42], which is a software framework for appearance based tracking. It includes all the components of LOFT-Appearance tracking with re-factored functionality and an easy plugin based C++ interface that includes several optional modules such as motion dynamics and a tiled image reader. A track-before-detect approach is employed which greatly reduces the search space and is handy especially in large WAMI imagery where objects look similar and have a very small support map. Constraining the search region also results in faster image read throughputs as only multi-threaded partial tiles are read in memory. LOFT-Lite can achieve processing rates of 300 milliseconds per frame (wall-clock time) per target. Large single camera WAMI frames JPEG compressed occupy anywhere near 25-28 Megabits and at 5 frames a second, the minimum throughput required is high enough to consider a large bandwidth streaming pipeline.

Figure 8.3: Illustrative example of data ecosystem: Tiled TIFF aerial image with a resolution of 7800x10600 pixels and ≈80 MB size. The zoomed up insets show the location of the objects that were tracked (right inset) in relation to the Bank of Albuquerque towers (left inset) with zoomed up views [107].

8.7 Cloud/Fog System Architecture

Figure 8.4 shows the cloud and fog architecture, which consists of three layers: Mobile User Layer, Fog Computation Layer, and Cloud Management Layer. The Mobile User Layer is comprised of services that handle both the collection and consumption activities for the proposed system. Incident scene images and video data is collected using security cameras, civilian smart phones, and aerial perspectives and imported into the system for transfer to the Fog Computation Layer. The processed visual information can be accessed at the consumption sites of users via thin-clients such as web browsers with interfaces to explore the outputs, or application client software that downloads the data for local exploration, or appliances that use protocols such as VNC, RDP or PCoIP to access virtual desktops with the exploration software. The consumption fogs could also host caching services to bring the processed data closer

to the user thin-clients and reduce the need to have round-trip requests to the cloud. It is possible that the consumption phase involving an expert analyst may result in active use of the caching services that leads to repost of data to the Fog Computation Layer for further processing as part of deep exploration activities.

In the Fog Computation Layer, one service manages the small instance processing in conjunction with directives from the Unified Resource Broker (URB) in the Cloud Management Layer, and another service acts as the gateway to move data from the fog to the cloud via a high-performance network overlay setup with SDN. Thus, the Fog Computation Layer transforms the public cloud infrastructure into a mobile cloud infrastructure and allows the management services in the public cloud to seamlessly operate close to the user collection or consumption sites for end-to-end orchestration and dynamic control of data processing locations. At the Cloud Management Layer, the scalable computing services as well as the URB orchestrate the computation placement either in the fog or in the cloud infrastructure. The URB serves as the brain of the cloud, and manages the dynamic distribution of the application processing workload to meet application QoS and user QoE requirements.

8.8 Regional-Scale Evaluation

Herein, we first consider characterize the resultant impact on the LOFT-Lite application compute offloading to the cloud when using multiple video resolutions corresponding to different mobile devices and under disaster network degradation conditions. Next, we show user QoE improvements in data throughput and tracking time

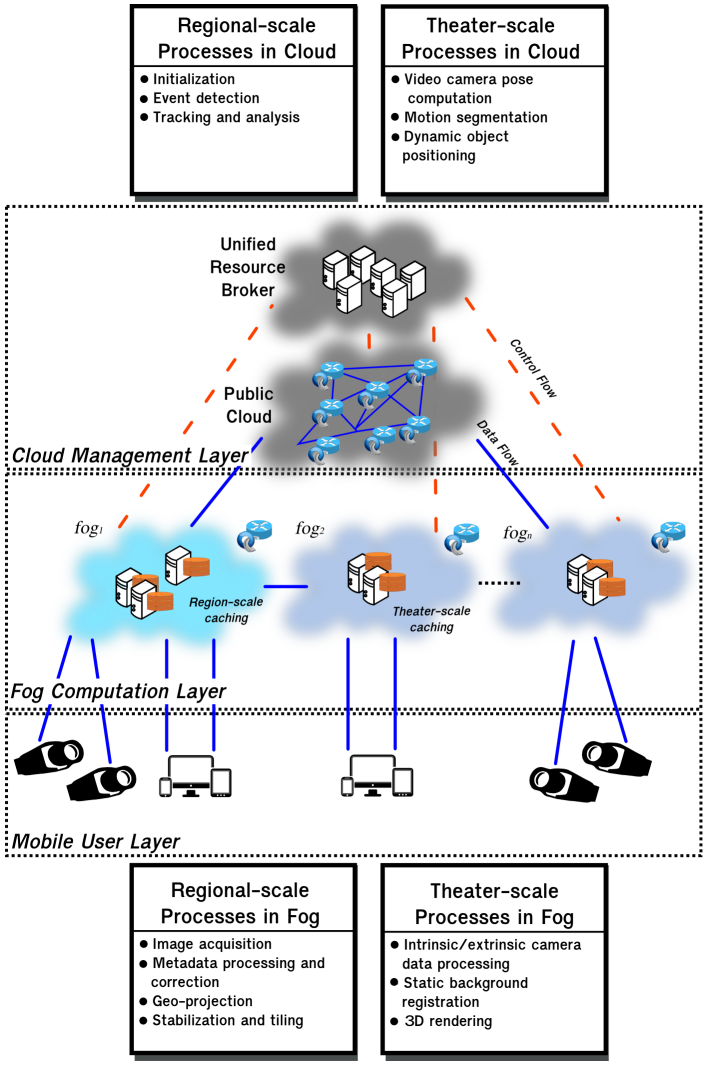
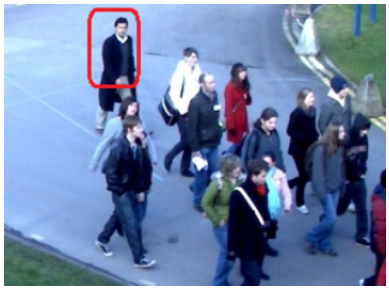


Figure 8.4: Illustration of the 3C system showing the relationships between mobile user, fog computation, and cloud management layers. The URB (Unified Resources Broker) controls how resources are provisioned and how data flows are routed with SDN between fogs and the public cloud. The small and large instance processing in the fogs and cloud for theater-scale and regional-scale applications is also shown [107].

when using our URB implementation that utilizes SDN and divides the LOFT-Lite application into small and large instance processing for cloud/fog computation, versus complete compute offloading to a core cloud over best-effort IP networks.

(a)



(b)

Figure 8.5: LOFT-Lite results on (a) standard and (b) Full-Motion surveillance video. Each frame in these video datasets is about 2MB compressed [107].

Disaster Network Experiments Setup and Results

Multiple video resolutions in practice need to be processed because the input source imagery in surveillance typically spans a wide variety sensor technologies found in mobile devices. In our experiments, we consider common resolutions in surveillance video belonging to the broad categories of: (a) Full-resolution WAMI (7800 x 10600) (see Figure 8.3), (b) Large-scale aerial video (2560 x 1900), and (c) Ground surveillance video (640 x 480) (see Figure 8.5). To consider disaster network scenarios systematically that impact data transfer, we assume a 4G-LTE network configuration with an initial bandwidth of 100 Mbps (best case) and apply a bandwidth degradation profile during compute offloading test cases with different resolutions. For experimental

purposes, the profile degrades the bandwidth at a rate of 20 Mbps per minute due to heavy cross-traffic load or candidate network path failures till it falls to zero (i.e., worst case disconnection scenario).

Our visual cloud computing setup for the disaster network experiments includes two virtual machines (VMs) for the data collection and computation sites, respectively each with a single core CPU and 1GB of main memory in a GENI platform testbed connected through an OpenFlow switch. Several performance metrics such as estimated throughout, tracking time, waiting time and total time are measured to characterize Quality of Application (QoA) of LOFT-Lite application computation as well as SCP (standard secure copy utility) data movement under the bandwidth degradation profile.

Table 8.1 shows measurement results averaged over ten trials with 95% confidence intervals. Our full-resolution WAMI and large-scale aerial video processing pipelines are non real-time and suffer relatively long wait times in comparison with the lower resolution ground-based FMV pipeline that runs in real-time. These results quantify system scalability and the benefits of reducing video resolution under disaster network conditions to support single target real-time tracking for multiple instances of LOFT-Lite. Standard video resolution results in the highest throughput over 3G/4G networks.

Table 8.1: QoA impact for compute offloading of multiple video resolutions for a systematic network degradation profile.

Performance Metrics	Full-resolution WAMI (7800 x 10600)	Large-scale aerial video (2560 x 1900)	Ground-based standard video (640 x 480)
(SCP QoA) Number of transferred frames	25.80 ± 0.26	180.9 ± 0.9	892 ± 9
(LOFT-Lite QoA) Estimated throughput (Mbps)	66.5 ± 0.9	76 ± 0.9	43.9 ± 0.4
(LOFT-Lite QoA) Tracking time (sec/fr)	0.4035 ± 0.005	0.368 ± 0.002	0.403 ± 0.004
(LOFT-Lite QoA) Waiting time (sec/fr)	9.03 ± 0.13	0.845 ± 0.014	0 ± 0
(LOFT-Lite QoA) Total time (sec/fr)	9.46 ± 0.13	1.214 ± 0.014	0.403 ± 0.004

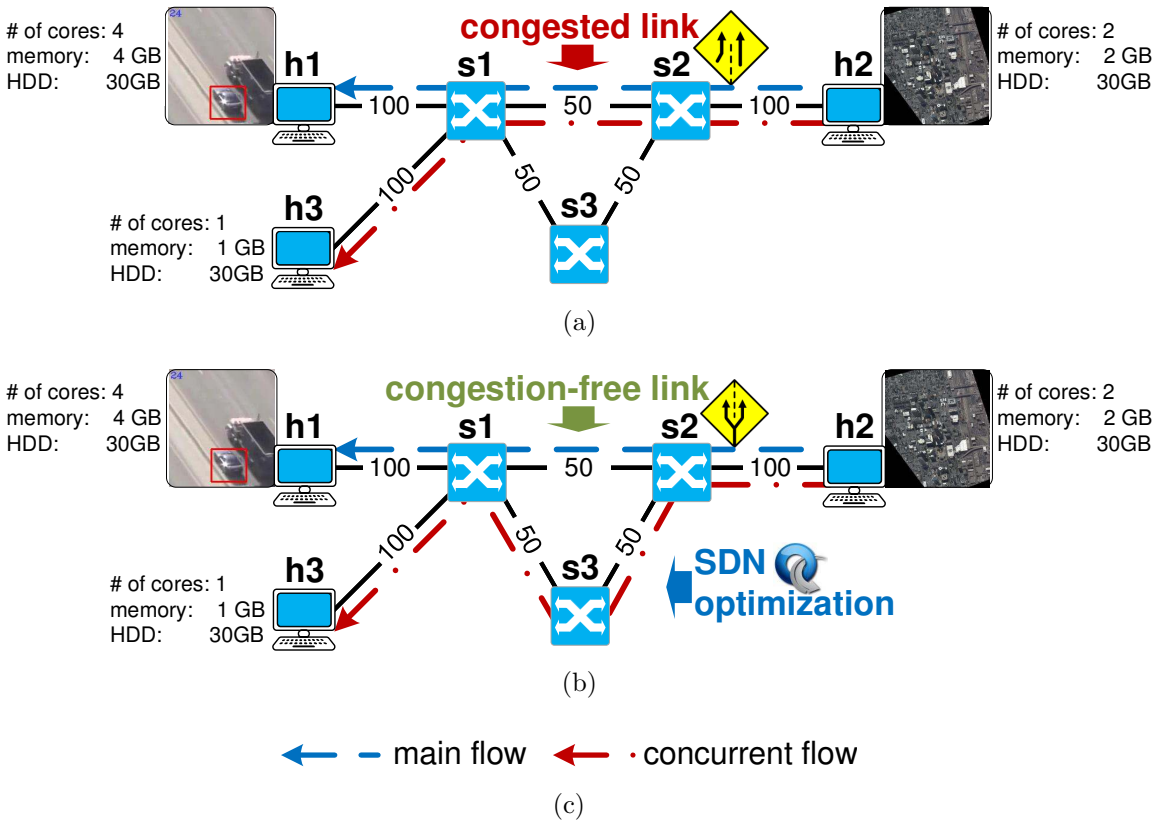


Figure 8.6: Data flows in the allocated GENI topology: (a) Standard video data flow interferes with concurrent flow on the $s2 \rightarrow s1$ link as regular network sends data through the best (the shortest) path; (b) Using SDN and the NPP algorithm, we optimize network resources usage and redirect concurrent flow through the longer path $s2 \rightarrow s3 \rightarrow s1$ which avoids congestion. Further, moving image pre-processing to the fog ($h2$ instead of $h1$) enables real-time tracking using LOFT-Lite [107].

Table 8.2: QoA impact results comparison for core cloud computing over IP network versus utilizing SDN and cloud/fog computing by dividing the application into small and large instance processing.

Performance Metrics	Core Cloud Computing over IP network	Cloud/Fog Computing utilizing SDN	Perceived Benefits
(SCP QoA) Storage transfer time (sec/fr)	0.564 ± 0.007	0.402 ± 0.006	Avoiding congestion with SDN traffic steering results in lower transfer time
(Imagemagick QoA) Pre-processing time (sec/fr)	0.1955 ± 0.0011	0.292 ± 0.023	No significant difference
(LOFT-Lite QoA) Estimated throughput (Mbps)	13.50 ± 0.34	41.85 ± 0.24	Lower transfer time and fog computation maximizes application throughput
(LOFT-Lite QoA) Tracking time (sec/fr)	0.4097 ± 0.0022	0.4229 ± 0.0024	No significant difference
(LOFT-Lite QoA) Waiting time (sec/fr)	0.902 ± 0.032	0 ± 0	Achieving maximum application throughput avoids waiting time and supports real-time computation
(LOFT-Lite QoA) Total time (sec/fr)	1.312 ± 0.034	0.4229 ± 0.0024	Cloud/fog computation of small and large instances can produce 3X speedup over core cloud computation

Cloud/Fog Computation Experiments Setup and Results

Standard (VGA) video resolution was used for the cloud/fog experiments to track pedestrians [130] in a crowd (see Figure 8.5b). An adaptive contrast enhancement global image pre-processing operation is applied as needed in the cloud/fog (using Imagemagick) before images are sent to the core cloud for object tracking. All images are pyramidal tiled TIFF (Tagged Image File Format) and the pre-processing retains the tile geometry.

Our setup for the cloud/fog computation experiments includes six virtual machines (VMs) in the GENI platform testbed as shown in Figure 8.6, where three of these VMs emulate OpenFlow switches ($s1$, $s2$ and $s3$) and others are regular hosts ($h1$, $h2$ and $h3$). Each host-to-switch link has 100 Mbps bandwidth, and each switch-to-switch link has only 50 Mbps bandwidth to emulate congested and damaged network infrastructure in a disaster scenario. Our LOFT-Lite application runs on $h1$ (quad-core CPU, 4GB of RAM and 30GB of HDD) which acts as a computation cloud site, whereas $h2$ (double-core CPU, 2GB of RAM and 30GB of HDD) acts as a collection fog site, and $h3$ (single-core CPU, 1GB of RAM and 30GB of HDD) consumes raw data from $h2$ by acting as a storage consumption fog site. Node $h3$ is configured with cross-traffic flow consumption such that it interferes with the main data traffic for the LOFT-Lite application. We call this cross-traffic as the 'concurrent flow', and the application traffic for LOFT-Lite as the 'main flow'. Finally, the thin-client (local PC) acts as a data consumer at the user end. LOFT-Lite runs on a thread with a backoff timer which sleeps for a specified delay while querying the local folder for the image stream. To transfer data between hosts, we use the SCP utility.

To differentiate between the cloud/fog and the core cloud computation, our experiment workflow is as follows: (i) start sending concurrent traffic from $h2$ to $h3$; (ii) start sending main traffic (video) from $h2$ to $h1$; (ii.a) while performing cloud/fog computing, start pre-processing concurrently with step (ii) (we assume here that pre-processing is faster than data transfer); (iii) wait till at least the first frame has been transferred; (iii.b) in case of core cloud computing, start pre-processing before step (iv) (in this case LOFT-Lite has to wait for each frame when its pre-processing ends); (iv) start LOFT-Lite; (v) wait until all main traffic has been transferred; and (vi) terminate both the applications and data transfers.

Table 8.2 shows the final timing results averaged over ten trials to estimate 95% confidence intervals for the cloud/fog and core cloud computation cases. For each trial, we used a 500 frame video sequence and measured several QoA performance metrics such as estimated throughput, tracking time, waiting time and total time. We can pre-process frames faster in the core cloud computation case in comparison to cloud/fog computation. Due to congestion in best-effort IP network and the unavailability of video at the computation cloud site, we cannot track with LOFT-Lite application in real-time (with 0 waiting time) in the core cloud computation case. Whereas in the cloud/fog computation utilizing SDN, LOFT-Lite can be run in real time at $3 - 4$ Hz.

Chapter 9

Conclusion and Future Research

The research presented in this dissertation showcases an appearance based tracking framework that is highly flexible and addresses common problems in difficult low resolution and low framerate video along with promising results in full motion video. We have proposed a novel adaptive appearance scheme that extends likelihood based matching in order to maintain a model that is free of occlusions and one that avoids drift. Automated tracking is an essential step for many applications such as incident detection, security, surveillance and also relatively newer applications like augmented reality. As of this writing almost 300 hours of video is uploaded to YouTube every minute. Newer higher resolution sensors and inexpensive storage has led to a large number of video being stored more and more every year. Automated analysis of such videos would require efficient tracking methodologies that are both, flexible and robust. The common theme throughout this work is to identify and isolate data specific challenges and tackle them in a way that can be treated as a plug and play

solution. The fusion framework along with demonstration of the integration work with many different engineering systems has shown great potential in terms of quality of results. We believe that our current system is flexible enough to continue more research where additional features or modules can contribute to better tracking.

9.1 Future directions

We recognize that several small changes can lead to improved results such as computing our histogram features on different parts of the foreground. Background can also be modeled with our current framework which would encode a more powerful part of the descriptor for more complex videos with dynamic backgrounds. Drift control can then be handled with the help of a background descriptor which would then fall under the class of discriminative classification. The flexible nature of LOFT allows researchers to add and test such potential modules or functionality in a rapid manner. As part of the natural extension to 2D tracking, incorporating 3D information also needs to be studied in more detail. Our initial set of results show a lot of promise for using 3D models of buildings to filter out short, mostly false, tracks. 3D information can assist directly in effective detection of occlusions and avoid computing the features on such frames. Even though current research mainly shows effective detection of tracks, our methodology along with comprehensive and accurate groundtruth allows for a more in depth analysis of the trajectories. Object tracking will always be a very complex task as data specific challenges will exist in the future. Keeping a flexible framework that can be adapted to such different conditions and to effectively

use and fuse different sources of information can bring tracking to a level where it can be as pervasive as image filters.

Bibliography

[1] K. Palaniappan, R. Rao, and G. Seetharaman. "Wide-area persistent airborne video: Architecture and challenges". In: *Distributed Video Sensor Networks* (2011), pp. 349–371.

[2] K. Palaniappan, F. Bunyak, P. Kumar, I. Ersoy, S. Jaeger, K. Ganguli, A. Haridas, J. Fraser, R. Rao, and G. Seetharaman. "Efficient Feature Extraction and Likelihood Fusion for Vehicle Tracking in Low Frame Rate Airborne Video". In: *IEEE Conference on Information Fusion (FUSION)* (2010), pp. 1–8.

[3] A. Haridas, R. Pelapur, J. Fraser, F. Bunyak, and K. Palaniappan. "Visualization of Automated and Manual Trajectories in Wide-Area Motion Imagery". In: *IEEE Conference on Information Visualization (IV)* (2011), pp. 288–293.

[4] J. Fraser, A. Haridas, G. Seetharaman, R. Rao, and K. Palaniappan. "KOLAM: An extensible cross-platform architecture for visualization and tracking in wide-area motion imagery". In: *SPIE Conference on Geospatial InfoFusion II (Defense, Security and Sensing: Sensor Data and Information Exploitation)* (2012), pp. 87470–87470.

[5] E. Blasch, P. Deignan, S. Dockstader, M. Pellechia, K. Palaniappan, and G. Seetharaman. "Contemporary Concerns in Geographical/Geospatial Information Systems (GIS) Processing". In: *IEEE National Aerospace and Electronics Conference (NAECON)* (2011), pp. 183–190.

[6] R. Porter, A. Fraser, and D. Hush. "Wide-area motion imagery". In: *IEEE Signal Processing Magazine* (2010), pp. 56–65.

[7] N. Cuntoor, A. Basharat, A. Perera, and A. Hoogs. "Track initialization in low frame rate and low resolution videos". In: *International Conference on Pattern Recognition* (2010), pp. 3640–3644. ISSN: 1051-4651.

[8] V. Reilly, H. Idrees, and M. Shah. "Detection and tracking of large number of targets in wide area surveillance". In: *IEEE European Conference on Computer Vision (ECCV)* (2010), pp. 186–199.

[9] H. Ling, Y. Wu, E. Blasch, G. Chen, H. Lang, and L. Bai. "Evaluation of Visual Tracking in Extremely Low Frame Rate Wide Area Motion Imagery". In: *IEEE International Conference on Information Fusion (FUSION)* (2011), pp. 1–8.

[10] J. Prokaj and G. Medioni. "Using 3D Scene Structure to Improve Tracking". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2011), pp. 1337–1344.

[11] E. Kao, M. Daggett, and M. Hurley. "An Information Theoretic Approach for Tracker Performance Evaluation". In: *IEEE Conference on Computer Vision (ICCV)* (2009), pp. 1523–1529.

[12] E. Pollard, A. Plyer, B. Pannetier, F. Champagnat, and G. Le-Besnerais. "GM-PHD filters for multi-object tracking in uncalibrated aerial videos". In: *IEEE Conference on Information Fusion (FUSION)* (2009), pp. 1171–1178.

[13] R. Porter, C. Ruggiero, and J. Morrison. "A framework for activity detection in wide-area motion imagery". In: *SPIE Conference on Defense, Security and Sensing* (2009), 73410O–73410O.

[14] G. Hager and P. Belhumeur. "Efficient region tracking with parametric models of geometry and illumination". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (1998), pp. 1025–1039.

[15] R. Kumar, H. Sawhney, S. Samarasekera, S. Hsu, T. Hai, G. Yanlin, K. Hanna, A. Pope, R. Wildes, D. Hirvonen, M. Hansen, and P. Burt. "Aerial video surveillance and exploitation". In: *Proceedings of the IEEE* (2001), pp. 1518–1539.

[16] Cootes T, G. Edwards, and C. Taylor. "Active appearance models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2001), pp. 681–685. ISSN: 0162-8828. DOI: 10.1109/34.927467.

[17] I. Matthews and S. Baker. "Active appearance models revisited". In: *International Journal of Computer Vision (IJCV)* (2004), pp. 135–164.

[18] R. Collins, Y. Liu, and M. Leordeanu. "Online Selection of Discriminative Tracking Features". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2005), pp. 1631–1643.

[19] S. Gu, Y. Zheng, and C. Tomasi. "Efficient visual object tracking with online nearest neighbor classifier". In: *Asian Conference on Computer Vision (ACCV)* (2010), pp. 271–282.

[20] B. Babenko, M. Yang, and S. Belongie. "Robust Object Tracking with Online Multiple Instance Learning". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2011), pp. 1619–1631.

[21] Z. Kalal, J. Matas, and K. Mikolajczyk. "P-N learning: Bootstrapping binary classifiers by structural constraints". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2010), pp. 49–56.

[22] I. Ersoy, K. Palaniappan, and G. Seetharaman. "Visual tracking with robust target localization". In: *IEEE Conference on Image Processing (ICIP)* (2012), pp. 1365–1368.

[23] X Mei, H. Ling, Y. Wu, and E. Blasch L. Bai. "Minimum error bounded efficient l_1 tracker with occlusion detection". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2011), pp. 1257–1264.

[24] X. Mei and H. Ling. "Robust Visual Tracking and Vehicle Classification via Sparse Representation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2011), pp. 2259–2272.

[25] Lu Zhang and Laurens van der Maaten. "Structure Preserving Object Tracking". In: *CVPR '13* ().

[26] X. Mei and H. Ling. "Robust visual tracking using *l1* minimization". In: *IEEE Conference on Computer Vision(ICCV)* (2009), pp. 1436–1443.

[27] T.F. Cootes and P. Kittipanya-Ngam. "Comparing variations on the active appearance model algorithm". In: *Proc. BMVC*. 2002, pp. 837–846.

[28] A. Adam, E. Rivlin, and I. Shimshoni. "Robust fragments-based tracking using the integral histogram". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2006, pp. 798–805.

[29] A. Smeulders, D. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. "Visual Tracking: An Experimental Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2014), pp. 1442–1468. ISSN: 0162-8828.

[30] Z. Kalal, K. Mikolajczyk, and J. Matas. "Tracking-learning-detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2012), pp. 1409–1422.

[31] S. Hare, A. Saffari, and P. Torr. "Struck: Structured output tracking with kernels". In: *IEEE Conference on Computer Vision (ICCV)* (2011), pp. 263–270.

[32] I. Saleemi and M. Shah. "Multiframe many–many point correspondence for vehicle tracking in high density wide area aerial videos". In: *International journal of computer vision (IJCV)* (2013), pp. 198–219.

[33] A. Perera, C. Srinivas, A. Hoogs, G. Brooksby, and W. Hu. "Multi-object tracking through simultaneous long occlusions and split-merge conditions". In: *Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2006, pp. 666–673.

[34] M. Keck, L. Galup, and C. Stauffer. "Real-time tracking of low-resolution vehicles for wide-area persistent surveillance". In: *Workshop on Applications of Computer Vision (WACV)*. IEEE. 2013, pp. 441–448.

[35] T. Pollard and M. Antone. "Detecting and tracking all moving objects in wide-area aerial video". In: *Computer Vision and Pattern Recognition Workshops (CVPR-W)*. IEEE. 2012, pp. 15–22.

[36] J. Prokaj, X. Zhao, and G. Medioni. "Tracking many vehicles in wide area aerial surveillance". In: *Computer Vision and Pattern Recognition Workshops (CVPR-W)*. IEEE. 2012, pp. 37–43.

[37] J. Xiao, H. Cheng, H. Sawhney, and F. Han. "Vehicle detection and tracking in wide field-of-view aerial video". In: *Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2010, pp. 679–684.

[38] Q. Yu and G. Medioni. "Motion pattern interpretation and detection for tracking moving vehicles in airborne video". In: *Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2009, pp. 2671–2678.

[39] Y. Huang and X. Luo. "Simultaneous detection and tracking in airborne video". In: *Computer Technology and Development (ICCTD)*. IEEE. 2009, pp. 320–324.

[40] Q. Li, Y. Yu, and R. Hou. "Real-time highway traffic information extraction based on airborne video". In: *Conference on Intelligent Transportation Systems*. IEEE. 2009, pp. 1–6.

[41] A. Basharat, M. Turek, Y. Xu, C. Atkins, D. Stoup, K. Fieldhouse, P. Tunison, and A. Hoogs. "Real-time multi-target tracking at 210 megapixels/second in wide area motion imagery". In: *Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2014, pp. 839–846.

[42] R. Pelapur, S. Candemir, F. Bunyak, M. Poostchi, G. Seetharaman, and K. Palaniappan. "Persistent target tracking using likelihood fusion in wide-area and full motion video sequences". In: *IEEE Conference on Information Fusion (FUSION)* (2012), pp. 2420–2427.

[43] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, et al. "The Visual Object Tracking VOT2015 challenge results". In: 2015. DOI: 10.1109/ICCVW.2015.79. URL: http://www.votchallenge.net/vot2015/program.html.

[44] Y. Wu, J. Lim, and M. Yang. "Online object tracking: A benchmark". In: *Computer Vision and Pattern Recognition (CVPR)*. 2013, pp. 2411–2418.

[45] Bruce D Lucas, Takeo Kanade, et al. "An iterative image registration technique with an application to stereo vision". In: *International Conf. on Artificial intelligence*. 1981.

[46] H. Nguyen and A. Smeulders. "Fast occluded object tracking by a robust appearance filter". In: *Pattern Analysis and Machine Intelligence (PAMI)* (2004), pp. 1099–1104.

[47] A. Adam, E. Rivlin, and I. Shimshoni. "Robust fragments-based tracking using the integral histogram". In: *Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2006, pp. 798–805.

[48] D.A. Ross, J. Lim, R.S. Lin, and M.H. Yang. "Incremental learning for robust visual tracking". In: *Int. J. Computer Vision* 77.1 (2008), pp. 125–141.

[49] H. Nguyen and A. Smeulders. "Robust tracking using foreground-background texture discrimination". In: *International Journal of Computer Vision (IJCV)* (2006), pp. 277–293.

[50] H. Bay, T. Tuytelaars, and L. Van Gool. "Surf: Speeded up robust features". In: *European conference on computer vision (ECCV)*. Springer. 2006, pp. 404–417.

[51] J. Kwon and M. Lee. "Tracking by sampling trackers". In: *International Conference on Computer Vision (ICCV)*. IEEE. 2011, pp. 1195–1202.

[52] Hyeonseob Nam and Bohyung Han. "Learning multi-domain convolutional neural networks for visual tracking". In: *arXiv preprint arXiv:1510.07945* (2015).

[53] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. "Learning spatially regularized correlation filters for visual tracking". In: *International Conference on Computer Vision (ICCV)*. 2015, pp. 4310–4318.

[54] J. Lee and W. Yu. "Visual tracking by partition-based histogram backprojection and maximum support criteria". In: *Internation Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2011, pp. 2860–2865.

[55] A. Hafiane, G. Seetharaman, K. Palaniappan, and B. Zavidovique. "Rotationally invariant hashing of median patterns for texture classification". In: *Lecture Notes in Computer Science (LNCS)* (2008), pp. 619–629.

[56] N. Dalal and B. Triggs. “Histograms of oriented gradients for human detection”. In: *Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2005.

[57] S. Nath and K. Palaniappan. “Adaptive robust structure tensors for orientation estimation and image segmentation”. In: *Advances in Visual Computing*. Springer, 2005, pp. 445–453.

[58] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence(PAMI)* (2002), pp. 971–987.

[59] A. Hafiane, G. Seetharaman, and B. Zavidovique. “Median binary pattern for textures classification”. In: *Image Analysis and Recognition*. Springer, 2007, pp. 387–398.

[60] S. Candemir, K. Palaniappan, F. Bunyak, G. Seetharaman, and R. Rao. “Feature prominence-based weighting scheme for video tracking”. In: *8th Indian Conference on Computer Vision, Graphics and Image Processing*. 2012. DOI: 10.1145/2425333.2425358.

[61] Kittler J, M. Hatef, R. Duin, and J. Matas. “On Combining Classifiers”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (1998), pp. 226–239.

[62] Z. Yin, F. Porikli, and R. Collins. “Likelihood Map Fusion for Visual Object Tracking”. In: *IEEE Workshop on Applications of Computer Vision (WACV)* (2008), pp. 1–7.

[63] D. Lukas and T. Kanade. “An iterative image registration technique with an application to stereo vision”. In: *Interational Joint Conferences on Artificial Intelligence (IJCAI)* (1981).

[64] H. Grabner, M. Grabner, and H. Bischof. “Real-time tracking via on-line boosting”. In: *British Machine Vision Conference (BMVC)* (2006), pp. 47–56.

[65] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma. “Robust face recognition via sparse representation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2009), pp. 210–227.

[66] B. Liu, L. Yang, J. Huang, P. Meer, L. Gong, and C. Kulikowski. “Robust and fast collaborative tracking with two stage sparse optimization”. In: *IEEE European Conference on Computer Vision (ECCV)* (2010), pp. 624–637.

[67] R. Pelapur, K. Palaniappan, F. Bunyak, and G. Seetharaman. “Vehicle Orientation Estimation using Radon transform-based voting in aerial imagery”. In: *SPIE Conference on Geospatial InfoFusion III (Defense, Security and Sensing: Sensor Data and Information Exploitation)* (2012).

[68] S. Grossberg. “Competitive learning: From interactive activation to adaptive resonance”. In: *Journal on Cognitive science* (1987), pp. 23–63.

[69] F. Bunyak and K. Palaniappan. “Efficient segmentation using feature-based graph partitioning active contours”. In: *IEEE Conference on Computer Vision (ICCV)* (2009), pp. 873–880.

[70] F. Bunyak, K. Palaniappan, S. Nath, and G. Seetharaman. "Flux tensor constrained geodesic active contours with sensor fusion for persistent object tracking". In: *Journal of Multimedia* (2007), pp. 20–33.

[71] K. Jafari-Khouzani and H. Soltanian-Zadeh. "Radon transform orientation estimation for rotation invariant texture analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2005), pp. 1004–1008.

[72] R. Pelapur, K. Palaniappan, and G. Seetharaman. "Robust orientation and appearance adaptation for wide-area large format video object tracking". In: *IEEE Conference on Advanced Video and Signal based Surveillance (AVSS)* (2012), pp. 337–342.

[73] Y. Bar-Shalom, X. Rong Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. Wiley-Interscience, 2001.

[74] G. Welch and G. Bishop. *An introduction to the Kalman filter*. Tech. rep. University of North Carolina, Chapel Hill, 1995.

[75] Peter Aundal Toft and John Aasted Sørensen. "The Radon transform-theory and implementation". PhD thesis. Technical University of Denmark, Department of Informatics and Mathematical Modeling, 1996.

[76] Jian Li, Shaohua Kevin Zhou, and Rama Chellappa. "Appearance modeling using a geometric transform". In: *IEEE Transactions on Image Processing* (2009), pp. 889–902.

[77] R. Pelapur, K. Palaniappan, and G. Seetharaman. "Robust Orientation and Appearance Adaptation for Wide-Area Large Format Video Object Tracking".

In: *IEEE Conference on Advanced Video and Signal based Surveillance (AVSS)* (2012), pp. 337–342.

[78] Air Force Research Laboratory. "COLUMBUS LARGE IMAGE FORMAT (CLIF) 2007 DATASET". In: *https://www.sdms.afrl.af.mil/datasets/clif2007/* ().

[79] T. Pock, R. Beichel, and H. Bischof. "A novel robust tube detection filter for 3d centerline extraction". In: *Image Analysis* (2005), pp. 55–94.

[80] B. Babenko, M.H. Yang, and S. Belongie. "Visual tracking with online multiple instance learning". In: *Proc. IEEE CVPR*. 2009, pp. 983–990.

[81] F. Bunyak, K. Palaniappan, S. K. Nath, and G. Seetharaman. "Geodesic active contour based fusion of visible and infrared video for persistent object tracking". In: *8th IEEE Workshop Applications of Computer Vision (WACV 2007)*. Austin, TX, 2007, Online.

[82] I. Ersoy, K. Palaniappan, R. Rao, and G. Seetharaman. "Tracking in persistent wide-area motion imagery". In: *Proc. SPIE Conf. Geospatial InfoFusion II (Defense, Security and Sensing: Sensor Data and Information Exploitation)*. Vol. 8396. 2012.

[83] M.J. Black and A.D. Jepson. "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation". In: *Int. J. Computer Vision* 26.1 (1998), pp. 63–84.

[84] A. Frangi, W. Niessen, K. Vincken, and M. Viergever. "Multiscale vessel enhancement filtering". In: *Medical Image Computing and Computer-Assisted Interventation (MICCAI)* (1998), pp. 130–137.

[85] T. Lindeberg and D. Fagerström. "Scale-space with casual time direction". In: *IEEE European Conference on Computer Vision (ECCV)* (1996), pp. 229–240.

[86] R. Hammoud, C. Sahin, E. Blasch, B. Rhodes, and T. Wang. "Automatic association of chats and video tracks for activity learning and recognition in aerial video surveillance". In: Multidisciplinary Digital Publishing Institute, 2014, pp. 19843–19860.

[87] W. R. Thissell, R. Czajkowski, F. Schrenk, T. Selway, A. J. Ries, S. Patel, P. L. McDermott, R. Moten, R. Rudnicki, G. Seetharaman, I. Ersoy, and K. Palaniappan. "A scalable architecture for operational FMV exploitation (AVAA)". In: 2015. DOI: 10.1109/ICCVW.2015.139. URL: https://sites.google.com/site/lsvsiccv2015/video-summarization-for-large-scale-analytics-workshop.

[88] A. Basharat, R. Pelapur, I. Ersoy, K. Palaniappan, and A. Hoogs. "Multi-Target Tracking in Video with Adaptive Integration of Appearance and Motion Models". In: *AIPR*. IEEE. 2016, submitted.

[89] P. McDermott, B. Plott, A. Ries, J. Touryan, M. Barnes, and K. Schweitzer. *Advanced Video Activity Analytics (AVAA): Human Factors Evaluation*. Tech. rep. DTIC Document, 2015.

[90] W. Thissell, R. Czajkowski, F. Schrenk, T. Selway, A. Ries, S. Patel, P. McDermott, and R. Rudnicki R. Moten, G. Seetharaman, et al. "A Scalable Architecture for Operational FMV Exploitation". In: *International Conference on Computer Vision Workshops(ICCV-W)*. 2015, pp. 10–18.

[91] D. Lowe. "Distinctive image features from scale-invariant keypoints". In: *Interational Journal of Computer Vision (IJCV)* (2004), pp. 91–110.

[92] O. Mendoza-Schrock, J. Patrick, and E. Blasch. "Video image registration evaluation for a layered sensing environment". In: *IEEE National Aerospace and Electronics Conference (NAECON)* (2009), pp. 223–230.

[93] G. Seetharaman, G. Gasperas, and K. Palaniappan. "A piecewise affine model for image registration in 3-D motion analysis". In: *IEEE Conference on Image Processing (ICIP)* (2000), pp. 561–564.

[94] A. Hafiane, K. Palaniappan, and G. Seetharaman. "UAV-Video registration using block-based features". In: *IEEE Conference on Geoscience and Remote Sensing Symposium (IGARSS)* (2008), pp. 1104–1107.

[95] K. Palaniappan and J. Fraser. "Multiresolution tiling for interactive viewing of large datasets". In: *17th Int. AMS Conf. on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography and Hydrology AMS Conference on Interactive Information and Processing Systems for Meteorology, Oceanography and Hydrology (IIPS)* (2001), pp. 338–342.

[96] C. Liu, W. Freeman, E. Adelson, and Y. Weiss. "Human-assisted motion annotation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2008), pp. 1–8.

[97] S. Candemir, K. Palaniappan, F. Bunyak, and G. Seetharaman. "Feature Fusion Using Ranking for Object Tracking in Aerial Imagery". In: *SPIE Conference on Geospatial InfoFusion II (Defense, Security and Sensing: Sensor Data and Information Exploitation)* (2012).

[98] E. Maggio and A. Cavallaro. *Video Tracking: Theory and Practice*. Wiley, 2011.

[99] D. Chen and J. Yang. "Robust Object Tracking via Online Dynamics Spatial Bias Appearance Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2007), pp. 2157–2169.

[100] Y. Wu, B. Wu, J. Liu, and H. Lu. "Probabilistic Tracking on Riemannian Manifolds". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2008), pp. 1–4.

[101] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet. "Color-Based Probabilistic Tracking". In: *IEEE European Conference on Computer Vision (ECCV)* (2002), pp. 661–675.

[102] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. "PROST Parallel Robust Online Simple Tracking". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2010), pp. 723–730.

[103] H. Grabner and H. Bischof. “On-line boosting and vision”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2006), pp. 260–267.

[104] B. Leininger, J. Edwards, J. Antoniades, D. Chester, D. Haas, E. Liu, M. Stevens, C. Gershfield, M. Braun, J. Targove, et al. “Autonomous real-time ground ubiquitous surveillance-imaging system (ARGUS-IS)”. In: *SPIE Defense and Security Symposium*. International Society for Optics and Photonics. 2008, 69810H–69810H.

[105] Eran Swears, Anthony Hoogs, and Kim Boyer. “Pyramid Coding for Functional Scene Element Recognition in Video Scenes”. In: *ICCV*. IEEE. 2013, pp. 345–352.

[106] K. Fieldhouse, M. Leotta, A. Basharat, R. Blue, D. Stoup, C. Atkins, L. Sherrill, B. Boeckel, P. Tunison, J. Becker, et al. “KWIVER: An open source cross-platform video exploitation framework”. In: *AIPR*. IEEE. 2014, pp. 1–4. URL: http://www.kwiver.org.

[107] R. Gargees, B. Morago, R. Pelapur, D. Chemodanov, P. Calyam, Z. Oraibi, Y. Duan, G. Seetharaman, and K. Palaniappan. “Incident-supporting visual cloud computing utilizing software-defined networking”. In: *IEEE Transactions on Circuits and Systems for Video Technology* (2016). DOI: 10.1109/TCSVT.2016.2564898.

[108] A. Herrera. “NVIDIA GRID: Graphics Accelerated VDI with the Visual Performance of a Workstation”. In: *NVIDIA Corp* (2014).

[109] A. Hossain. "Framework for a cloud-based multimedia surveillance system". In: *International Journal of Distributed Sensor Networks* (2014).

[110] L. Zhou and H. Wang. "Toward blind scheduling in mobile media cloud: Fairness, simplicity, and asymptotic optimality". In: *IEEE Transactions on Multimedia* (2013), pp. 735–746.

[111] Y. Wen, X. Zhu, J. Rodrigues, and C. Chen. "Cloud mobile media: reflections and outlook". In: *IEEE Transactions on Multimedia* (2014), pp. 885–902.

[112] W. Zhu, C. Luo, J. Wang, and S. Li. "Multimedia cloud computing". In: *IEEE Signal Processing Magazine* (2011), pp. 59–69.

[113] B. Liu, Y. Chen, R. Blasch, K. Pham, D. Shen, and G. Chen. "A holistic cloud-enabled robotics system for real-time video tracking application". In: *Lecture Notes in Electrical Engineering* (2014), pp. 455–468.

[114] B. Liu, Y. Chen, A. Hadiks, E. Blasch, A. Aved, D. Shen, and G. Chen. "Information fusion in a cloud computing era: a systems-level perspective". In: *IEEE Aerospace and Electronic Systems Magazine (AES-M)* (2014), pp. 16–24.

[115] H. Agrawal, C. Mathialagan, Y. Goyal, N. Chavali, P. Banik, A. Mohapatra, A. Osman, and D. Batra. "CloudCV: Large Scale Distributed Computer Vision as a Cloud Service". In: *arXiv:1506.04130* (2015).

[116] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. "OpenFlow: Enabling Innovation in Campus

Networks". In: *ACM Computer Communication Review (CCR)* (2008), pp. 69–74.

[117] S. Seetharam, P. Calyam, and T. Beyene. "ADON: Application-Driven Overlay Network-as-a-Service for Data-Intensive Science". In: *IEEE Conference on Cloud Networking (CloudNet)* (2014), pp. 313–319.

[118] *OpenFlow Switch Specification*. https://www.opennetworking.org/sdn-resources/openflow/57-sdn-resources/onf-specifications/openflow.

[119] M. Berman, J. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar. "GENI: A federated testbed for innovative network experiments". In: *Computer Networks* (2014), pp. 5–23. ISSN: 1389-1286.

[120] Z. Jiang, D. Chan, M. Prabhu, P. Natarajan, H. Hao, and F. Bonomi. "Improving Web Sites Performance Using Edge Servers in Fog Computing Architecture". In: *IEEE Symposium on Service Oriented System Engineering (SOSE)* (2013), pp. 320–323.

[121] A. Mohammad. and H. Eui-Nam. "Fog Computing micro datacenter based dynamic resource estimation and pricing model for IoT". In: *IEEE Conference on Advanced Information Networking and Applications (AINA)* (2015), pp. 687–694. ISSN: 1550-445X.

[122] I. Stojmenovic and W. Sheng. "The Fog computing paradigm: Scenarios and security issues". In: *IEEE Conference on Computer Science and Information Systems (FedCSIS)* (2014), pp. 1–8.

[123] N. Truong, L. Gyu, and Y. Ghamri-Doudane. “Software defined networking-based vehicular Adhoc Network with Fog Computing”. In: *IEEE Symposium on Integrated Network Management (IM)* (2015), pp. 1202–1207.

[124] E. Chemeritskiy and R. Smeliansky. “On QoS management in SDN by multi-path routing”. In: *Conference on Science and Technology (MoNeTeC)* (2014), pp. 1–6.

[125] H. Egilmez and A. Tekalp. “Distributed QoS architectures for multimedia streaming over software defined networks”. In: *IEEE Transactions on Multimedia (MM)* (2014), pp. 1597–1609.

[126] Y. Tsung-Feng, K. Wang, and Y. Hsu. “Adaptive routing for video streaming with QoS support over SDN networks”. In: *Conference on Information Networking (ICOIN)* (2015), pp. 318–323.

[127] K. Govindarajan, K. Meng, H. Ong, M. Wong, S. Sivanand, and S. Leong. “Realizing the quality of service (QoS) in Software-Defined Networking (SDN) based cloud infrastructure”. In: *Conference on Information and Communication Technology (ICoICT)* (2014), pp. 505–510.

[128] P. Calyam, S. Rajagopalan, A. Selvadhurai, S. Mohan, A. Venkataraman, A. Berryman, and R. Ramnath. “Leveraging OpenFlow for Resource Placement of Virtual Desktop Cloud Applications”. In: *IEEE Symposium on Integrated Network Management (IM)* (2013), pp. 311–319.

[129] H. Aliakbarpour, K. Palaniappan, and G. Seetharaman. “Robust camera pose refinement and rapid SfM for multiview aerial imagery Without RANSAC”.

In: *IEEE Geoscience and Remote Sensing Letters (GRSL)* (2015), pp. 2203–2207.

[130] A. Ellis, A. Shahrokni, and J. Ferryman. "Pets2009 and winter-pets 2009 results: A combined evaluation". In: *IEEE Workshop on Performance Evaluation of Tracking and Surveillance (PETS-Winter)*. 2009, pp. 1–8.

VITA

Rengarajan Pelapur was born in Chennai, India. After graduating from Dr. Kalamadi Shamarao High School in Pune, India he attended Maharashtra Academy of Engineering under the University of Pune where he received his B.S. degree in 2010. In fall of 2010, he joined the CIVA lab working on projects funded by ARL and AFRL. He also held intern positions at Kitware during Summer 2012 and PerceptiMed during Spring 2016. His current research interests include image processing, computer vision, and pattern recognition with emphasis on wide area persistent surveillance, target tracking, segmentation for quantification of vascular structures in epi-fluroscence imagery. He has reviewed for several IEEE international journals and conferences. He received the Ph.D. degree in computer science from the University of Missouri-Columbia in 2016.