UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

EXTENDED OBSERVATION PARTICLE FILTER WITH SVD TEMPLATE GENERATION IMPLEMENTED FOR GPU

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

JONATHAN DAVID WILLIAMS
Norman, Oklahoma
2018

EXTENDED OBSERVATION PARTICLE FILTER WITH SVD TEMPLATE GENERATION IMPLEMENTED FOR GPU

A DISSERTATION APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

This dissertation is dedicated to my parents

Cynthia Maria Avery and Fred Travis Avery,

to Judy Williams, to the memory of Arlen Williams,

and to the memory of my grandparents.

Acknowledgements

I would like to thank:

Dr. Joseph Havlicek for his infinite patience and assistance preparing this work. My fellow researchers and close friends Dr. Nick Mould for convincing me to persue a Ph.D. and Dr. Chuong Nguyen for sharing his expansive knowledge and expertise.

Table of Contents

List of Tables

List of Figures

Abstract

EXTENDED OBSERVATION PARTICLE FILTER WITH SVD TEMPLATE GENERATION IMPLEMENTED FOR GPU

Jonathan David Williams, Ph.D.
The University of Oklahoma, 2018

Supervisor: Joseph P. Havlicek

This work presents a novel template updating strategy based on singular value decomposition (SVD) together with an expansion and extension of previous work combining observations across temporally adjacent frames to implement a likelihood function that provides improvement to velocity refinement in a particle filter tracker. SVD as a novel approach to template generation is used to take advantage of the intuitive notion that the largest singular value corresponds to the highest correlate across template candidates which should more adequately represent the target appearance while rejecting noise and other distractions for use in a correlation based scoring system such as the proposed likelihood function extended across temporally adjacent frames. The tracker is implemented to accelerate computationally expensive operations by moving them to the GPU for processing. This proposed expanded likelihood function provides an improvement of 11.2%-12.5% to particle degeneracy as compared to the previous method in the "Augmented State Vector" approach [4] across the composite of videos. This improvement to particle degeneracy provides for a lower requirement in the number of actual particles necessary for implementation of a particle filter tracker and thus a lower computational requirement while simultaneously providing similar performance. The proposed SVD template generation provides 23.8% increase in time before track loss when comparing the best case in each category of update-by-score and update-by-SVD across the composite of videos. While not bench-marked in this work for quantitative comparison, the use of the GPU with Tensorflow-GPU and Python has allowed the large data set needed for analysis to be obtained in days instead of the months that would have been required in my original proof-of-concept work that was targeted solely for CPU in Matlab.

Chapter 1

Introduction

There are many applications in which the ability to track moving targets in video signals provides intriguing opportunities when automated by computer. Historically there are situations which have required a human in the loop such as in video surveillance. Automating these tasks requires overcoming problems such as noisy video and changing target appearance that humans don't intuitively pay much attention to, but that computers have to be able to accommodate.

There are many options available to engineers when confronted with the task of automated target tracking. A handful of these options and a general background of the field will be presented in Chapter 2. Particle filter based tracking is one such option that faces numerous obstacles which will be discussed. Template updating strategies face many issues that lead to detrimental performance for visual trackers. Particle filtering, which is plagued by problems known as particle degeneracy and particle impoverishment, is a method that can be employed in a target tracker which will be discussed in greater detail in Chapter 3. It is the intention of this work to confront the problems seen in template updating and particle degeneracy and add more robust options for those seeking to use particle filter based target tracking.

The proposed techniques as well as a detailed experimental setup for quantitative assessment of them will be presented in Chapter 3. Particle filtering has many computations that can be performed in parallel. It is therefore beneficial that the potential for GPU acceleration of particle filter trackers be examined and the hardware and software platforms chosen will be discussed here.

The results of the proposed methods are promising. SVD template generation has provided a time before track loss improvement of up to 25.6% allowing for more robust tracking. Extension of the importance function across 3 frames has provided an improvement of 12.5% to particle degeneracy providing for more robust tracking and an option for lower computational cost. A detailed analysis of the experiment with focus on the improvements to more robust tracking and better tracking error will be presented and discussed in Chapter 4.

Lastly this work will conclude and present potential future research areas in 5.

The main original contributions of this dissertation are as follows:

1. Extended likelihood function for reduction of particle degeneracy.

2. SVD template generation for more robust appearance tracking.

3. GPU accelerated implementation by novel use of Tensorflow-GPU and Python.

Chapter 2

Target Tracking Review

Tracking targets is a visual task that is important in myriad applications, both commercial and military. Novel applications of target tracking include biometric identification, surveillance, video content analysis, and autonomous modes for vehicles [36]. The task of tracking targets in video can be divided into the subcategories of detection, tracking, and behavior analysis [18, 36]. Each of these categories are faced with their own unique challenges and are the subject of intense research today. Detection is the process of identifying targets within a photo or video. Tracking is the process of identifying the motion of the targets detected. Behavior analysis is the topic of discovering intent in the tracked targets once their motions are discovered and is useful for purposes of surveillance in security, law-enforcement, and military applications.

2.1 Target Detection

Detection is the task of identifying candidate targets within a video and is, by modest estimate, currently the most intensely researched and broadest of these subcategories due to advances in computational hardware capabilities. These advances hold the promise to enable the practical application of deep learning and sparse representation techniques [29, 33] that were considered impractical merely a decade ago.

2.1.1 Segmentation

Most tracking algorithms first identify potential target regions of interest (ROI) for further analysis by means of image segmentation or other techniques such as linear or nonlinear filtering. Potential target ROI detection can be performed by segmenting the image in various ways such as mean-shift [6] or normalized cuts [36] as seen in Figure 2.1. Mean-shift can be applied to cluster image pixels on a multitude of criteria such as pixel intensity, hue, or by texture statistics in a local region around the pixel such as by intensity histogram. Mean-shift is a process that iteratively clusters pixels by shifting them to a mean value of nearby pixels based on the chosen criterion. The process ends when pixels no longer shift by an appreciable threshold. Each pixel cluster, when reflected back into the original position of the pixels before mean-shifting, forms an image segment.

Identification of ROIs in more recent development of the You-Only-Look-Once (YOLO) algorithm [25, 26] using convolutional neural networks (CNN) takes advantage of the input stage of a CNN by adoption reuse of the convolutional computation for use in image segmentation and determination of ROIs. This input stage consists of a bank of filters that apply various kernels such as Gabor or wavelet kernels in preparation for input into a pre trained deep learning neural network. Many of these kernels inherently enhance edges and connected regions of similar texture. Figure 2.2 shows how the implied input of convolutional filters (left) are reused for calculation of ROIs as bounding boxes (top) while classification is performed in the normal way of a CNN (bottom). The result is simultaneous determination of bounding box ROI and classification (right).

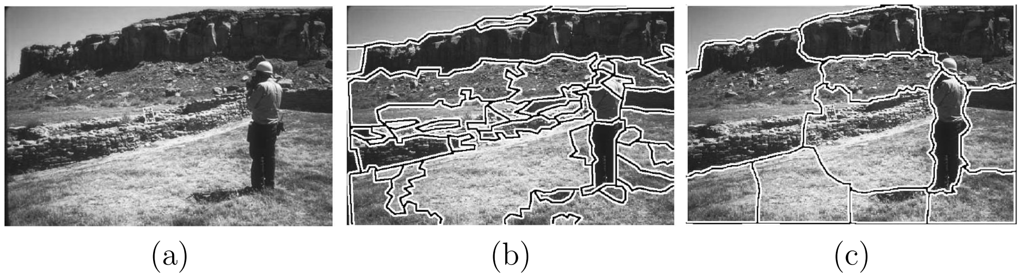


Figure 2.1: Segmentation examples [36]©ACM 2006: (a) original image, (b) segmentation by mean-shift, (c) segmentation by normalized cuts

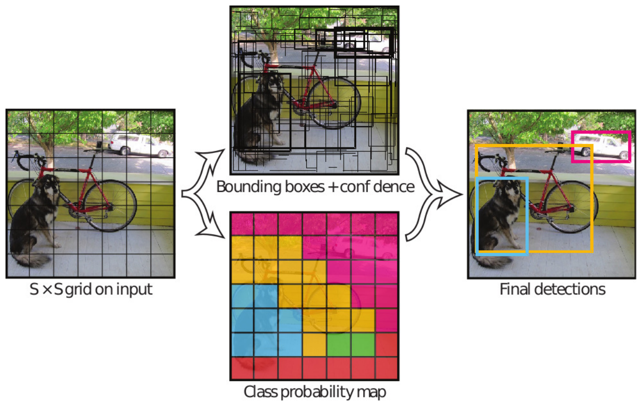


Figure 2.2: Example of segmentation and classification, the YOLO algorithm [25]©CoRR 2015. Segmentation by possible bounding boxes(top-center), classification by CNN (bottom-center).

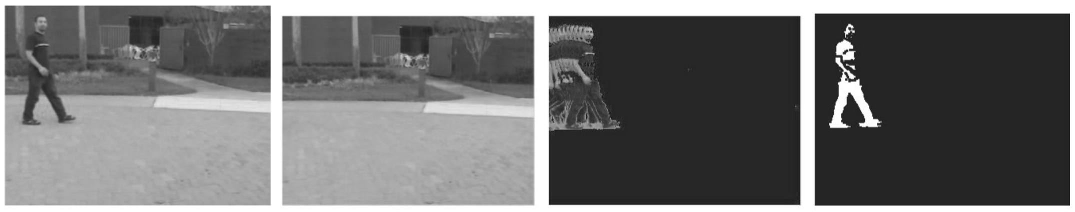
The detection of potetential targets can also be implemented by considering the inverse problem of identifying what is background and bringing the compliment of that forward as potential targets before classification [36]. In this way potential targets can be identified by simply being different from background. We can see examples of this in Figure 2.3 where Gaussian mixture modeling is used to model the background for target extraction, and in Figure 2.4 that uses an eigenspace decomposition to represent the background as a subspace in image space from which foreground targets are identified by their distance from the background's eigenspace subspace [36].

Segmentation is followed by further analysis of each ROI for the purposes of classification of those regions as target, not target, or for individual isolation in the case of multiple target scenarios by a multitude of possible techniques [2, 3, 6, 14, 23, 28, 33].

2.1.2 Identification and Classification

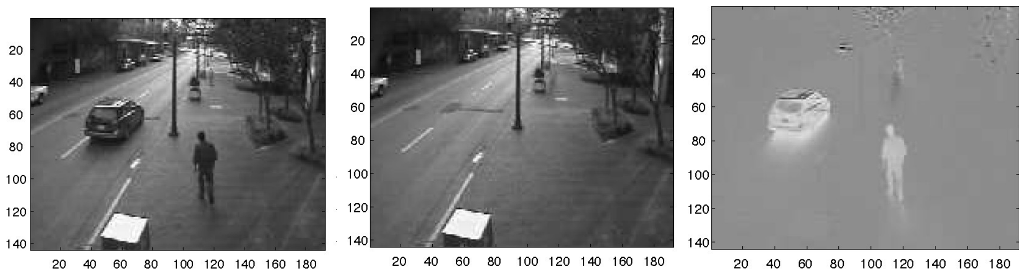
Target classification techniques are varied, and can range from using something as simple as a pixel-based template or a histogram of intensities to something more complex such as a sparse representation or a pre-trained convolutional neural net [14, 25, 26, 37] as in the YOLO algorithm seen in Figure 2.2, or a pre-trained heirarchy of features [33]. Targets can also be identified uniquely from frame to frame by use of an algorithm based on target outline features or contours as shown in Figure 2.5 [36].

Identifying a target in most cases requires the use of some kind of visual model of potential target or targets. These models vary widely and should be evaluated for their applicability in any given tracking problem [29]. Target



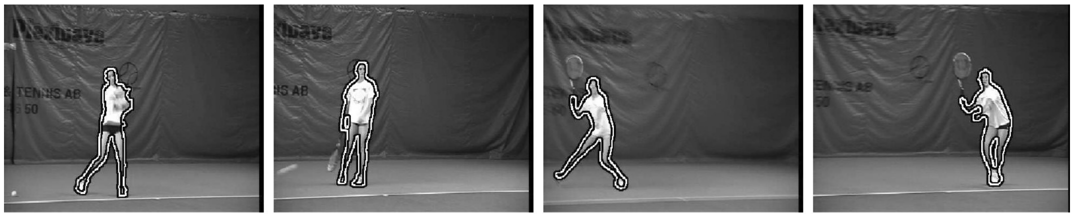
(a) (b) (c) (d)

Figure 2.3: Background modeling for extracting potential targets using Mixture of Gaussians [36]©ACM 2006: (a) image from video of pedestrian, (b) mean of Gaussians of highest weight, (c) mean of Gaussians of second highest weight, (d) result of background subtraction

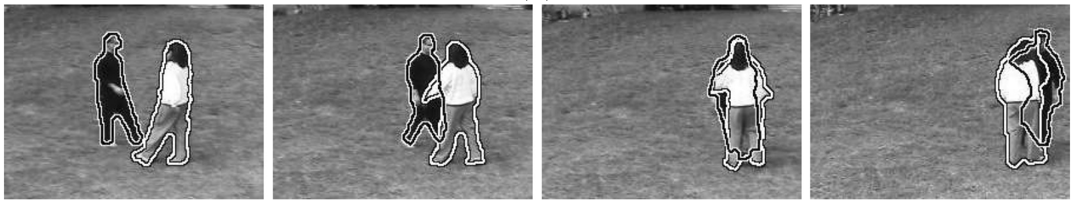


(a) (b) (c)

Figure 2.4: Background modeling based on Eigenspace decomposition [36] ©ACM 2006: (a) image from video of vehicle with pedestrian, (b) result of projecting frame on background Eigenspace, (c) difference image between (b) and (a)



(a)



(b)

Figure 2.5: Contour based target tracking [36]©ACM 2006: (a) single target, (b) multiple targets with occlusion.

models can consist of features [33] such as conjoined geometric shapes, outlines and contours, textures, statistics of textures [7, 23], offline or online learning models based on support vector machines or deep learning [14, 37], sparse representation as used in the matching-pursuit techniques [2], or any combination of these [29, 36]. Identified targets are pared down and isolated subsequently by use of kinematics or statistical models of potential target motion to build the resulting track [13, 18].

Template Update Issues

Target appearance can change dynamically due to a host of different factors including changes in aspect and range arising from relative motion between the sensor and target, noise, intermittent glare or shadow, or occlusion. More drastic changes in appearance can arise when the target is a deformable body. One familiar example is human perambulation which is often handled by employing an articulated target model [20]. This requires a visual target model to also be dynamic in order to accommodate. Target appearance changes and updating the model presents yet another type of tracking problem when the appearance model is considered another form of state that must be tracked and updated through a kind of visual state space.

There is also a problem that arises when parts of the background behind a target are captured in an hypothesized target region during update. This phenomenon known as background-leakage is often cumulative and can result in an appearance model that includes so much background that track loss occurs due to the tracking algorithm locking onto background instead of target [20]. This is especially problematic in appearance models that comprise only an image of the target as demonstrated in Figure 2.6. In the figure, the target template can be seen in the upper-left corner. From these templates, it can be seen that the road and some buildings leak into the template over time. Many of the solutions to this problem involve the use of an exhaustive search in the form of a convex optimization problem as can be seen in the Lucas-Kanade algorithm [3, 20].

Simply determining when a template update is necessary can be fraught with challenges unique to the choice of visual model. One of the simplest techniques is to force an update every frame using the current observation [20] or on a regular time interval. More nuanced techniques make use of a mechanism to determine when the target observation is sufficiently different from the target appearance model. One such technique utilizes the discriminatory capabilities of multiple AM-FM feature channels to determine when the model has significant drift from the observed target [28]. In the matching pursuit (MP) algorithms, template updating occurs when a new target observation is

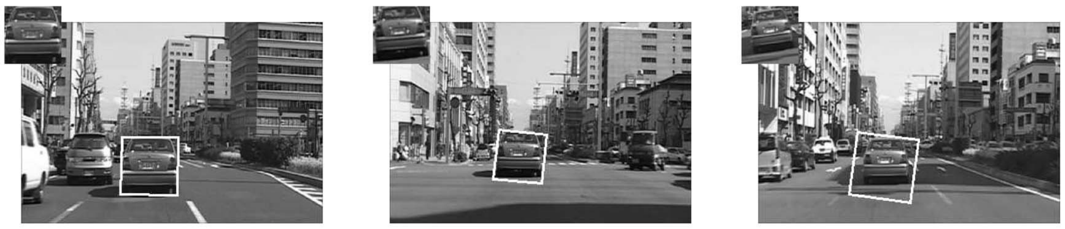


Figure 2.6: Background leakage example [20]©IEEE 2004

sufficiently different from those in its model by its distance from the visual space spanned by a dictionary of observations [2]. In the same vein a pixel based template update can be triggered by a mechanism that determines that significant drift in appearance has occurred by use of the distance between the observation and a visual eigen-space generated from a set of observed templates not unlike the dictionary used in MP algorithms [12].

2.2 Motion Tracking

Target tracking is the task of predicting target movement in order to associate detected targets from one frame to the next. Targets motion can be sewn together most simply by proximal location. Tracking the motion can also take advantage of known target kinematics in order to make predictions about target movement [18]. Detailed knowledge of target kinematics allows for identified targets to be linked frame to frame and isolated from other potential targets and background.

2.2.1 Bayesian Filtering

Under appropriate conditions, Bayesian filtering can be used to accurately estimate the target state [1, 10, 30]. Bayesian filtering consists of estimating the posterior probability density (posterior pdf) of the state $\mathbf{x}_t$ conditioned on the incoming observations $\mathbf{z}_{1:t}$ over all previous times, resulting in the conditional probability density [1]

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}). \tag{2.1}$$

A deterministic model for the change in the system at each time step is known a priori and is variously called the state update equation, the dynamic equation, or the state transition equation [1, 10, 18, 31]; it is given by

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{v}_{t-1}), \tag{2.2}$$

where $\mathbf{f}$ is a function of the state $\mathbf{x}_{t-1}$ and an i.i.d. process noise $\mathbf{v}_{t-1}$ at a previous time index.

Application of this update equation to a state pdf results in a new pdf conditioned on the previous one, and this is described by the conditional pdf [1]

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}). \tag{2.3}$$

The observation or measurement equation which must also be known a priori describes the relationship between the state and what can be observed. It is given by [1]

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t, \mathbf{n}_t), \tag{2.4}$$

where $\mathbf{x}_t$ is the current state and $\mathbf{n}_t$ is i.i.d. observation noise.

Assuming that the state update depends only on the previous state, that is, that the process is first-order Markov, one can write the update equation conditioned on the set of all observations as [1]

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})d\mathbf{x}_{t-1}. \tag{2.5}$$

At each new time step the desired density function (2.1) can be calculated using Bayes' rule resulting in the equation [1]

$$p(\mathbf{x}|\mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})} \tag{2.6}$$

where the denominator is simply a normalizing constant to ensure the resulting pdf integrates to one [1]. The output of the filter is simply the expected value of this density.

Direct use of these equations requires very precise knowledge of the system under observation including the update function, the measurement function, and characteristics of the process and measurement noise. Example optimal solutions mentioned in the Arulampalam tutorial on particle filters [1] are the Kalman Filter, Extended Kalman Filter (EKF), and grid based methods. Their use requires exacting knowledge of the system, however, and alternative techniques must be utilized for more general cases where the system under investigation might be non-linear or their process and observation noises non-Gaussian as in the use case of the EKF, or the states are not finite and discrete as in the case of grid based methods.

2.2.2 Kalman Filter

When the state vector posterior pdf, process noise, and observation noise are known to be or are assumed to be jointly Gaussian, then these densities can be represented parametrically and an optimal solution exists known as a Kalman filter [15]. Let Q_{t-1} and $\mathbf{n}_t$ be the covariance matrices of the process noise $\mathbf{v}_{t-1}$ and measurement noise $\mathbf{n}_t$, respectively. Let the notation $\mathcal{N}(\mathbf{x}; \mathbf{m}, P)$ indicate that x is Gaussian distributed with mean $\mathbf{m}$ and covariance P as in [15]. If the system is linear, then the state trajectory $\mathbf{x}_t$ can be modeled by the state update equation [15]

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{v}_{t-1}) = F_t \mathbf{x}_{t-1} + \mathbf{v}_{t-1}, \tag{2.7}$$

where F_t is the state update matrix at time t and

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t, \mathbf{n}_t) = H_t \mathbf{x}_t + \mathbf{n}_t \tag{2.8}$$

is the observation. In (2.8), H_t is the observation matrix at time t. The matrices F_t and H_t must be known for all t. Then from the Bayesian filtering equations (2.5) and (2.6) can be derived a set of recurrence relationships for the conditional pdfs using linearity and established knowledge of Gaussian pdf behavior [15]:

$$p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}) \sim \mathcal{N}(\mathbf{x}_{t-1}; \mathbf{m}_{t-1|t-1}, P_{t-1|t-1}), \tag{2.9}$$

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) \sim \mathcal{N}(\mathbf{x}_t; \mathbf{m}_{t|t-1}, P_{t|t-1}), \tag{2.10}$$

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) \sim \mathcal{N}(\mathbf{x}_t; \mathbf{m}_{t|t}, P_{t|t}), \tag{2.11}$$

where

$$\mathbf{m}_{t|t-1} = F_t \mathbf{m}_{t-1|t-1}, \tag{2.12}$$

$$P_{t|t-1} = Q_{t-1} + F_t P_{t-1|t-1} F_t^T, \tag{2.13}$$

$$\mathbf{m}_{t|t} = \mathbf{m}_{t|t-1} + K_t(\mathbf{z}_t - H_t \mathbf{m}_{t|t-1}), \tag{2.14}$$

$$P_{t|t} = P_{t|t-1} - K_t H_t P_{t|t-1}, \tag{2.15}$$

and the innovation equation is

$$S_t = H_t P_{t|t-1} H_t^T + R_t. \tag{2.16}$$

The so-called "Kalman gain" in (2.14) and (2.15) is then given by [15]

$$K_t = P_{t|t-1} H_t^T S_t^{-1}. \tag{2.17}$$

2.2.3 Extended Kalman Filter

When the state equation (2.2) and measurement function (2.4) are non-linear, a sub-optimal solution can be obtained by linearizing them about an operating point x to perform what is known as first-order extended Kalman filtering (EKF). This allows the same methods used in the Kalman filter to be used recursively as before, but with substitutions:

$$\hat{F}_t = \left.\frac{d\mathbf{f}_t(\mathbf{x})}{d\mathbf{x}}\right|_{\mathbf{x}=\mathbf{m}_{t-1|t-1}} \tag{2.18}$$

in place of F in the equation (2.13) and

$$\hat{H}_t = \left.\frac{d\mathbf{h}_t(\mathbf{x})}{d\mathbf{x}}\right|_{\mathbf{x}=\mathbf{m}_{t|t-1}} \tag{2.19}$$

in place of H in the equation (2.15) as linearizations about the appropriate operating points at the mean values $\mathbf{m}_{t-1|t-1}$ and $\mathbf{m}_{t|t-1}$, respectively. The purpose of this is to allow for the reflection of the Gaussian pdfs as before through the process and observation functions as if they were linear and well behaved operations that provide Gaussians from Gaussians through the linear transformations. The Arulampalam tutorial paper notes that higher order EKFs can be obtained by expanding (2.18) and (2.19) in higher-order Taylor series; these are computationally expensive however [1].

2.2.4 SIS Particle Filter

Recursive Bayesian filters can be implemented using a technique known as particle filtering in order to overcome exceptionally difficult if not impossible to calculate integrals of their analytic form or to accommodate potentially non-linear functions or systems that have non-Gaussian process or measurement noises [1]. A particle filter does this by utilizing a Monte Carlo approach by representing the probability density functions with a collection of point masses distributed and weighted so as to adequately represent the underlying pdf. The previously mentioned integral (2.5) becomes a more computationally reasonable summation. The denominator normalization factor of (2.6) is simply the sum of the particle weights.

Filtering involves a two-step process making use of (2.3) to predict the prior (to conditioning on observation) density followed by the filtering step which updates the weights based on the concept of importance sampling to produce the posterior (to conditioning on observation) density. This two-step process is repeated sequentially for every incoming observation and has been named Sequential Importance Sampling (SIS) [1]. The importance density or observation function is a function that can be readily evaluated for a given state $\mathbf{x}$ and the incoming observation. This observation function is used to update the weights of particles on receipt of a new observation using the formula [1]

$$w_t^i \propto w_{t-1}^i \frac{p(\mathbf{z}_t|\mathbf{x}_t^i)p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i)}{q(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i, \mathbf{z}_t)}, \tag{2.20}$$

where $p(\mathbf{z}_t|\mathbf{x}_t^i)$ is the likelihood function, $p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i)$ is the prior density represented as point masses $i \in [1, N_p]$. In (2.20), the denominator $q(\cdot)$ is the proposal density, also known as the importance density or importance function. It is an assumed pdf from which the particles are drawn when the SIS filter is run [1]. Design of the proposal density $q(\cdot)$ is an important issue that can dramatically affect the filter performance. Performing computation of the importance function and weight update on each particle, while expensive in total, can be performed entirely in parallel.

2.2.5 Resampling

One of the major drawbacks to the use of SIS particle filtering is that over merely a few iterations the compounded system noise variance $\mathbf{v}_t$ can cause all of the particle weights except one to become negligibly small. When this occurs, the set of particles and weights no longer adequately represents the underlying pdf and the filter fails. This phenomenon is commonly named particle degeneracy and due to the inadequate pdf representation it results in an unacceptable error in the estimated value. Particle degeneracy is often measured using an approximate effective sample size N_{eff} given by [1]

$$N_{\mathit{eff}} = \frac{1}{\sum_{i=1}^{N_p} (w_t^i)^2}. \tag{2.21}$$

To mitigate the effect of particle degeneracy a new set of particles can be drawn from the existing ones in a technique known as resampling [1, 10, 31]. In resampling, the new set of particles includes an increased number of particles located in the state space where they more adequately represent the underlying pdf. Resampling algorithms are very diverse and are the subject of many textbooks devoted exclusively to their study [17, 19, 24, 27]. Researchers have even developed a novel resampling algorithm inspired by the gathering behavior of spider monkeys [27].

One of the most accessible pedagogical devices for conceptualizing and understanding multinomial, stratified, systematic, and residual resampling algorithms is the "wheel representation" given in [24] and depicted in Fig. 2.7. In this representation the wheels represent a cumulative distribution of weights around a wheel such that a value between zero and one lies at a point on each of these wheels, corresponding to a weight W and its corresponding particle. Multinomial resampling is performed for each particle in the new set by drawing a random number from zero to one and selecting the corresponding particle at that location on the wheel [24]. Stratified resampling is performed by first dividing the wheel into equal sections and choosing a random number that lies in each section. In this way, particles that span several sections are always chosen unlike multinomial resampling that maintains a probability, however small, that larger particles may not be selected. Systematic resampling is performed by generating a random number from zero to one, then taking the particles located at equal intervals around the wheel from that point. Residual resampling is a more complex process involving two steps [24]. These steps can be thought of in a way similar to long-division except that in this case the whole number increment is $\frac{1}{N}$, N being the number of particles. The first step draws particles that have larger weight than $\frac{1}{N}$ as many times as $\frac{1}{N}$ can fit within the weight of that particle. The remainder of each particle once the multiples of $\frac{1}{N}$ have been subtracted is then normalized as a whole before the second step. The second step is the same as multinomial resampling with these weight remainders.

In a particle filter, resampling is typically done when the N_{eff} calculation in (2.21) falls below a threshold to draw new particles before the prediction step that generates the prior. The process of resampling introduces another problem in that sometimes there are only a small number of identical particles remaining after resampling thereby also inadequately representing the pdf. This phenomenon is called particle impoverishment [1, 19].

Resampling is notorious for its need to include the entire set of particles and weights together, typically precluding the use of parallel execution. There

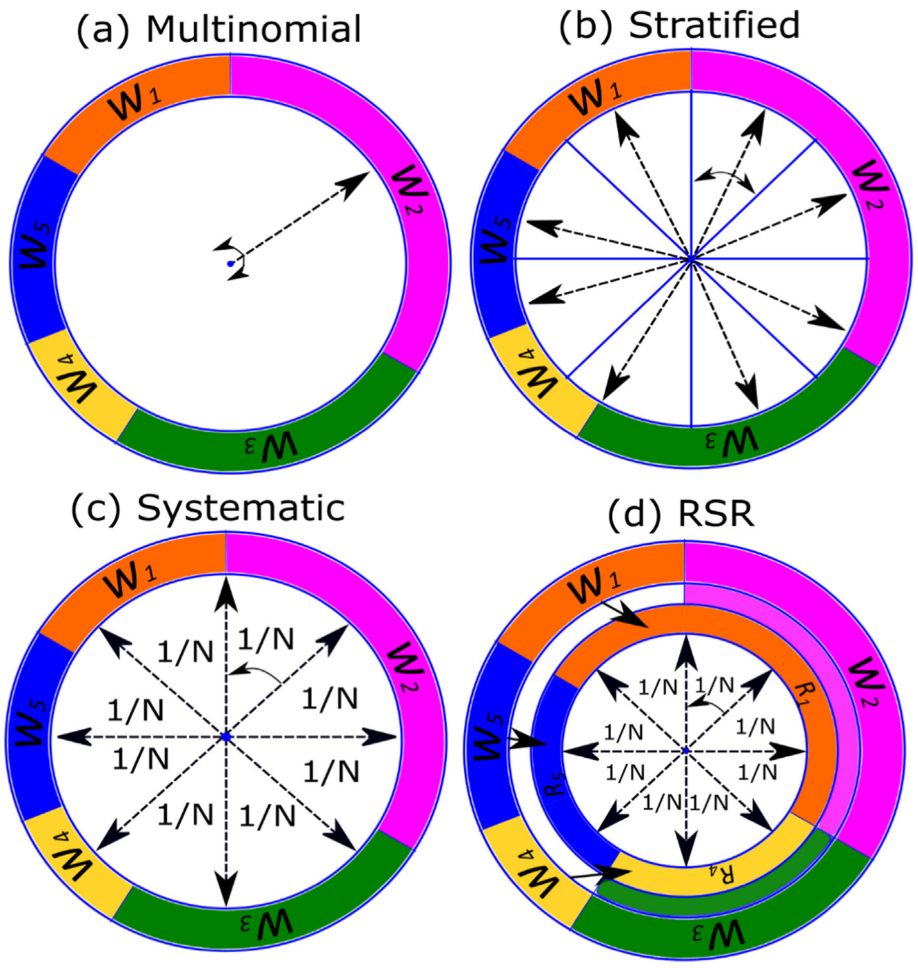


Figure 2.7: Wheel representation of resampling methods [24] ©Microelectronics Reliability 2018: (a) Multinomial, (b) Stratified, (c) Systematic, (d) Residual

have recently been developments intended to remove this limitation of the resampling step by application of Metropolis resampling on graphics processing unit (GPU) hardware [21].

2.2.6 Auxiliary Particle Filter

To lessen the effects of both particle degeneracy and particle impoverishment a resampling technique known as auxiliary particle resampling has been developed in what is called an auxiliary particle filter [1]. In auxiliary particle filtering a set of particles, called auxiliary particles, is generated in the same manner as the prior pdf in the prediction step, and these are also weighted in a similar fashion as the filtering set using the current observation. However, these new particles and weights are only used as a means to resample the state pdf of the previous time step on margin of the new observation before running the usual steps of the SIR particle filter. In this way it is like replacing the re-sampling step at the end of the SIR particle filter with a special auxiliary re-sampling step at the beginning of each cycle. By performing resampling in this way particles that have reasonable state representation as determined by the observation in the new frame are preemptively pared down thereby increasing the number of particles that land in good representational locations when generating the new prior. Mathematically speaking it is refining the prior pdf on margin of the new observation by means of the resampling mechanism.

2.3 Visual Tracking with Particle Filter

The likelihood functions used in a particle filter for visual tracking can be based on pixel-wise templates and correlation, histograms, shapes, and many others [29, 36]. Correlation based likelihood functions typically take the form of normalized cross correlation (NCC) between the expected target appearance and the corresponding spatial support hypothesized by a particle within the frame as follows [3]:

$$\rho_t^i = \frac{\sum \mathbf{z}_t \mathbf{z}_t^i}{\sqrt{\sum \mathbf{z}_t^2 \sum (\mathbf{z}_t^i)^2}}, \tag{2.22}$$

where $\mathbf{z}_t$ is the pixel template at time t biased such that the mean pixel value is zero and $\mathbf{z}_t^i$ is the ROI of a particle indexed by i also biased such that its mean pixel value is zero, and where the summation is taken over every pixel location in the template and ROI (indices implied by summation). The numerical range of NCC takes values from -1 (perfect anti-correlation) to 1 (perfect correlation) and as such are typically passed through another function to provide a gain and restrict the results to non-negative values needed to represent a pdf. One such function composition has a gain k and an exponential (2.23) that forces the resulting weight to always be greater than zero as follows [4]:

$$w_t^i \propto p(\mathbf{z}_t \mid \mathbf{x}_t^i) = e^{-k(1-\rho_t^i)}. \tag{2.23}$$

2.3.1 SIR Filter and Improvement of the Likelihood Function

The key to using particle filtering in any particular application is the design of an importance function that represents the pdf of the state vector to the best extent possible. Poor choice of importance functions can be either too loose or too tight in the state space, causing higher error in the former, and track loss in the latter. In worst case, the selected importance function may not represent the true state trajectory at all and performance of the particle filter will be extremely poor.

The sampling importance resampling (SIR) filter [1, 11] is a particle filtering algorithm variant that is widely used for visual target tracking. In the SIR filter, the importance function is set equal to the prior density of the state vector and resampling is performed at every time step. This implies that the particle weights are then directly proportional to the likelihood function [1]. Therefore, performance of the SIR filter is driven by the design of the likelihood function.

In many cases, the likelihood function may neglect or fail to consider certain state variables, particularly when those variables are unobservable. For example, velocity is commonly used as a state variable in visual target tracking filters. However, standard video cameras cannot provide any direct measurement of velocity. Consequently, visual target tracking filters typically include velocity as a state variable but almost universally omit the velocity from the likelihood function altogether. In the case of a correlation based particle filter, an improvement to provide an indirect measurement of velocity and incorporate it into the likelihood function was presented in [4]. It was shown that implementing a likelihood function that explicitly incorporated the previous observation together with the previous location of each particle could improve performance by providing an indirect measurement of velocity. The likelihood function used in [4] was given by (2.23) with the normalized cross correlation defined by

$$\hat{\rho}^i_{t,t-1} = \frac{\sum \mathbf{z}_t \mathbf{z}^i_t + \sum \mathbf{z}_{t-1} \breve{\mathbf{z}}^i_{t-1}}{\sqrt{\sum \mathbf{z}_t^2 + \sum \mathbf{z}_{t-1}^2}\sqrt{\sum \left(\mathbf{z}^i_t\right)^2 + \sum \left(\breve{\mathbf{z}}^i_{t-1}\right)^2},} \tag{2.24}$$

where as in the NCC formula (2.22), the terms are biased to mean zero and where $\mathbf{z}_{t-1}$ is the actual observation from the previous frame and $\breve{\mathbf{z}}^i_{t-1}$ is the target appearance hypothesized by particle i in the previously observed video frame from time $t-1$. Intuitively, (2.24) is the normalized cross correlation (NCC) between the observations and the hypothesis of particle $\hat{\mathbf{x}}_t^i$ calculated across two temporally adjacent frames, thereby providing a means to measure the velocity indirectly and incorporate it into the likelihood function explicitly.

2.4 Summary

Target tracking in video is a broad and vigorously investigated field with many potential applications. Finding appropriate models of highly dynamic target appearance is a significant open problem that includes difficult challenges such as determining when an appearance model update is needed and how it should be performed. Challenging problems inherent in updating a template from video observations include background leakage and dynamic appearance changes such as those caused by articulation; it is critical to overcome these problems in order to provide robust tracking performance. Target kinematics also must be modeled and the state of that model must be iteratively updated based on observations to be able to track a target across video frames in a robust way. The use of particle filtering with a kinematic model is faced with the problems of particle degeneracy, particle impoverishment, and the selection of an appropriate importance and likelihood functions. Visual tracking using only frames of video specifically has the unfortunate circumstance that it does not provide a direct observation of the velocity components of targets in order to accurately observe the state causing the state pdf to be inadequately pared down on margin of the incoming observations. This limitation in the observation has been effectively circumvented to some degree by performing the correlation calculations across adjacent frames and templates as described in [4].

Chapter 3

Problem Statement

Target tracking in video by means of particle filters is faced with several shortcomings. In the general case of target tracking it is often difficult to obtain an appropriate template from the source video [2, 3, 20, 23]. Background leakage causes the template to be contaminated with objects and textures from the background after which tracking error increases and tracking is eventually lost due to the tracker mistaking background for target. Particle filter based trackers face issues of particle degeneracy and particle impoverishment inherent in particle filters more generally [1, 4, 10, 27]. In the widely used SIR filter, design of the likelihood function is critically important for maintaining an appropriate set of particles by observation to adequately represent the pdf of the target state. Without an adequate observation into the elements of this target state to maintain the pdf around that elemental dimension, particle degeneracy generally occurs and substantially degrades the performance of the filter.

3.1 Novel SVD Template Generation

In particle filter based trackers it is common to update a template by identifying the particle across a window of frames that has the highest likelihood value as seen in the approximate maximum-likelihood (AML) method in [5]. This value is often influenced by partial occlusion, glare, shadow, or noise which causes the highest score to not be perfectly aligned with the target leading to the background leakage issue [20]. To address this issue, I propose a novel template generation strategy in which the high likelihood scoring template ROIs are combined by means of Singular Value Decomposition (SVD) as a means to reject outlying optical effects. The hypothesis here is that by application of the SVD the optical effects and background textures will tend to be delegated to the lowest singular values in the decomposition. SVD consists of decomposing a matrix into three matrices U, Σ, and V where U has unitary columns corresponding to individual diagonal entries of the diagonal matrix Σ that are the singular values and V is a matrix that can be intuitively though of as a mixing matrix that provides appropriate linear combinations of the matrix product of U and Σ to generate the original matrix T before decomposition. The highest singular value represents the highest correlate across the set of acquired potential targets making it together with its corresponding column in the matrix U ideal for use in generation of a composite template to be used as a correlation based likelihood function. To generate the composite template we collect an ROI corresponding to the particle evaluated with the highest likelihood function across all particles, one for every frame over a fixed window up to and including the current frame. Let T_k represent such a set for $k \in [t - w,\ t]$ where t is the current frame and w is the width of our window. A matrix T_j where $j \in [1,\ w]$ is produced by the column-wise assembly of the columnated elements of T_k as seen in Figure 3.1.

SVD is then performed on the T_j matrix such that

$$T = U \Sigma V^* \tag{3.1}$$

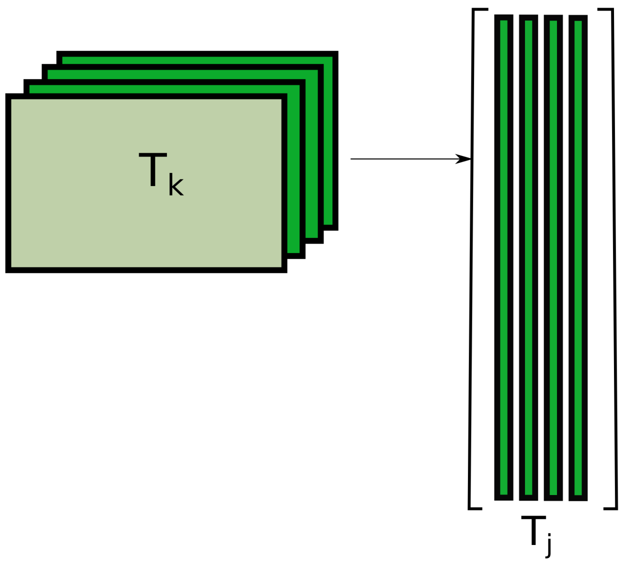


Figure 3.1: Generation of T_j from window of template candidates T_k

Let $U_{j;k}$, $\Sigma_{j;k}$, and $V_{j;k}$ represent the jth column and kth row of U, Σ, and V respectively. Then the composite template $\tilde{T}$ is generated by

$$\tilde{T} = reshape(U_{1;}\Sigma_{1;1}V_{1;1}), \tag{3.2}$$

where notation $U_{1;}$ represents the first column of U, and the reshape function reverses the effect of prior columnation of T_k to provide a 2D template. The use of $V_{1;1}$ ensures the resulting composite is not inverted in sign. Examples of the resulting SVD generated template can be seen in Figure 3.2.

3.2 Extended Likelihood Function

It was shown in section 2.3.1 that by incorporating the particle ROI across adjacent frames a likelihood function could be built that would improve the selectivity of the velocity components of particle states. It is the purpose of this research to determine if adding additional time frames to the likelihood

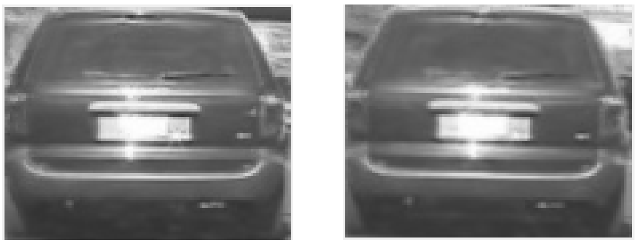

(a) (b)

Figure 3.2: Two examples of SVD template generation from the Car4 sequence: (a) after 20 frames, (b) after 100 frames

function would provide more improvement. Analysis of the quantitative effect on N_{eff} of the original technique and a new technique incorporating additional adjacent time frames will be provided to support the hypothesis that these techniques improve the particle degeneracy issue. The new likelihood function introduced incorporates observations and a slight change of notation as follows:

$$\hat{\rho}^i_{t,t-1,t-2} = \frac{\sum \mathbf{z}_t \mathbf{p}^i_t + \sum \mathbf{z}_{t-1} \mathbf{p}^i_{t-1} + \sum \mathbf{z}_{t-2} \mathbf{p}^i_{t-2}}{\sqrt{\sum \mathbf{z}^2_t + \sum \mathbf{z}^2_{t-1} + \sum \mathbf{z}^2_{t-2}} \sqrt{\sum \left(\mathbf{p}^i_t\right)^2 + \sum \left(\mathbf{p}^i_{t-1}\right)^2 + \sum \left(\mathbf{p}^i_{t-2}\right)^2}}, \tag{3.3}$$

where the $\mathbf{z}_t$ is the template from time t biased to mean zero and $\mathbf{p}^i_t$ is the ROI of the ith particle hypothesis at time t biased to mean zero making (3.3) a form of the normalized cross correlation (NCC) between the ROI of the hypothesis of a particle $\hat{\mathbf{x}}^i_t$ across three temporally adjacent frames and the corresponding template from those same time instances. The resulting $\hat{\rho}^i_{t,t-1,t-2}$ is then passed through

$$w^i_t \propto p(\mathbf{z}_t \mid \mathbf{x}^i_t) = e^{-k(1-\hat{\rho}^i_{t,t-1,t-2})} \tag{3.4}$$

to produce the resulting likelihood function.

3.3 GPU Accelerated Parallel Execution

One of the primary advantages to the use of particle filters is their inherent support for parallel execution. Recent developments in hardware and software targeted toward accelerating the training and execution of artificial neural networks are ripe for exploitation. The Tensorflow platform is one such advance in that it has been geared toward optimized use of graphics processor unit (GPU) and tensor processor unit (TPU) hardware yet presents itself intuitively as a way to build data-flow and processing diagrams in the form of Tensorflow graphs. It is in this vein that Tensorflow together with Python is be the platform of choice to provide hardware acceleration for this set of experiments. More specifically, Tensorflow has a built-in low-level functions for multiply-accumulate operations and a high-level operation available to perform the bilinear interpolation necessary to extract ROIs from incoming frames.

The Tensorflow graphs for this experiment have been automatically generated using TensorBoard and are cataloged in Appendix B. Likelihood functions for visual trackers often use relatively expensive cross correlation calculations and these likelihood functions can be evaluated entirely in parallel to much advantage. Figure B.4 shows the graph for obtaining the ROIs for every particle, the output of which is fed into the graph of Figure B.5 which embodies the likelihood function.

3.4 Experimental Test Parameters

The experiment includes ten runs of each combination of parameters discussed below and listed in Table 3.2 for a grand total of 57,600 runs in the set. The number of particles proportionally affects the amount of computation that the tracking algorithm uses and in that vein particle counts of 100, 300, 700 and 1000 are used to compare the effects and potential trade off between computational cost and performance.

3.4.1 System Dynamics Equation

The target state is defined in six dimensions as follows:

$$\mathbf{x}_t = [m, n, dm, dn, s, r]^T \tag{3.5}$$

where m and dm are respectively the vertical position and velocity, n and dn are respectively the horizontal position and velocity, s is the visual scaling factor, and r is the visual rotation factor in radians, and T is the vector transpose operation. The state update equation corresponding with Equation (2.2) is:

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{v}_{t-1}) = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{v}_{t-1} \tag{3.6}$$

where

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

is a constant velocity, magnification, and rotation model wherein $\mathbf{v}_t$ is i.i.d. process noise drawn from $\mathcal{N}(0, P)$ with the covariance $P{=}diag(0, 0, 2, 2, 0.05, 0.02)$. It was necessary to choose particularly large covariance values in order to accommodate the diversity of the selected videos.

3.4.2 Video Sequences

A diverse set of twenty videos was selected from Visual Tracker Benchmark project [34] (visible spectrum), Browncamp [22] (infrared), and AMCOM (infrared) [8, 9, 16, 32, 35, 36] sets to perform our experimental analysis against. These were selected to provide variety in resolution, target appearance, and target motion against which to test our proposed methods. The Visual Tracker Benchmark videos contain vehicular and human targets in various conditions of lighting, occlusion, and appearance changes. The Browncamp videos are fixed camera infrared videos of vehicles driving down a road. The AMCOM videos are aerial infrared videos of military vehicles performing maneuvers. Table 3.1 provides a detailed description of the selected videos individually.

3.4.3 Template Updating Comparisons

The set includes a traditional update by highest likelihood score and the proposed SVD generation update for comparison. The effect of using varying history depth for template selection and SVD generation is also observed. The effect of varying template update intervals is also observed.

3.4.4 Likelihood Function Comparisons

The likelihood functions are given the initials NCC, ASV, ASVHO for normalized cross correlation, 2-timestep correlation, and the proposed 3-timestep correlation respectively. These initials are inspired by the original work in [4], the "Augmented State Vector" and the proposed method "Augmented State Vector of Higher Order".

3.5 Experimental Hardware and Software Setup

The bulk of this experiment was performed on a system with the specifications listed in Table 3.3. A GUI was built using PyQt5 for running the experiment

Table 3.1: Video sequences with descriptions.

Car4	Van driving in sunlight and under a bridge
CarScale	SUV approaching camera with partial occlusion and changing orientation
Coke	Soda can with random acceleration, orientation, and full occlusion
Crossing	Fixed camera of pedestrian in cross walk
Dudek	Human head moving randomly as owner stands, sits, turns, changes facial expressions
Football	Football player helmet/head, multiple similar targets
Girl	Girl's head, owner in front of window spinning in an office chair
RedTeam	Low resolution, noisy, and long video of red vehicle navigating a course in open tundra
Skater	Low resolution Olympic ice skater performing maneuvers
Skater2	Low resolution Olympic ice skater in all black performing maneuvers
bc1_case3	Fixed camera of car driving away
bc3_case7	Fixed camera of SUV driving away partially obscured by following car
rng14_15	Truck, target is a small smudge
rng16_18	Tank, target is a small smudge
rng18_16	Truck, indistinguishable from background in first frame
rng19_06	Vehicle thermally obscured by clutter
rng19_13	Tank in a caravan turning and partially obscured by clutter
rng19_NS	Very short sequence of tank
walking	Fixed camera of pedestrian walking with partial occlusion by lamp post
woman	Woman walking with partial occlusion

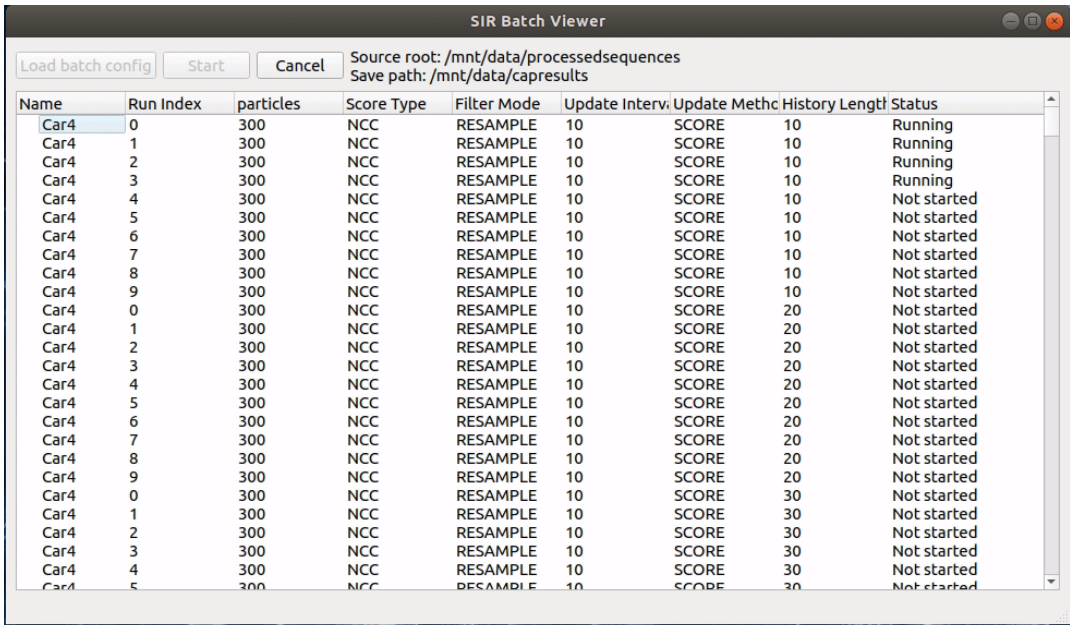
Table 3.2: Experiment Parameters

Particle Counts	100, 300, 700, 1000
Filter Type	Resample, Auxiliary
Template Update Methods	Best Score, SVD
Template History Depths(frames)	10, 20, 30
Template Update Intervals(frames)	10, 20
Likelihood Functions	NCC, ASV, ASVHO

Table 3.3: Hardware Specifications

CPU	Intel® Core™ i5-2550K CPU @ 3.40GHz x 4
Memory	16GB
GPU	Nvidia® GeForce™ GTX 1070
OS	Ubuntu 18.04.01 LTS 64-bit
Python	3.6.6
Tensorflow GPU	1.12.0
Qt	5.9.5
PyQt5	5.11.2

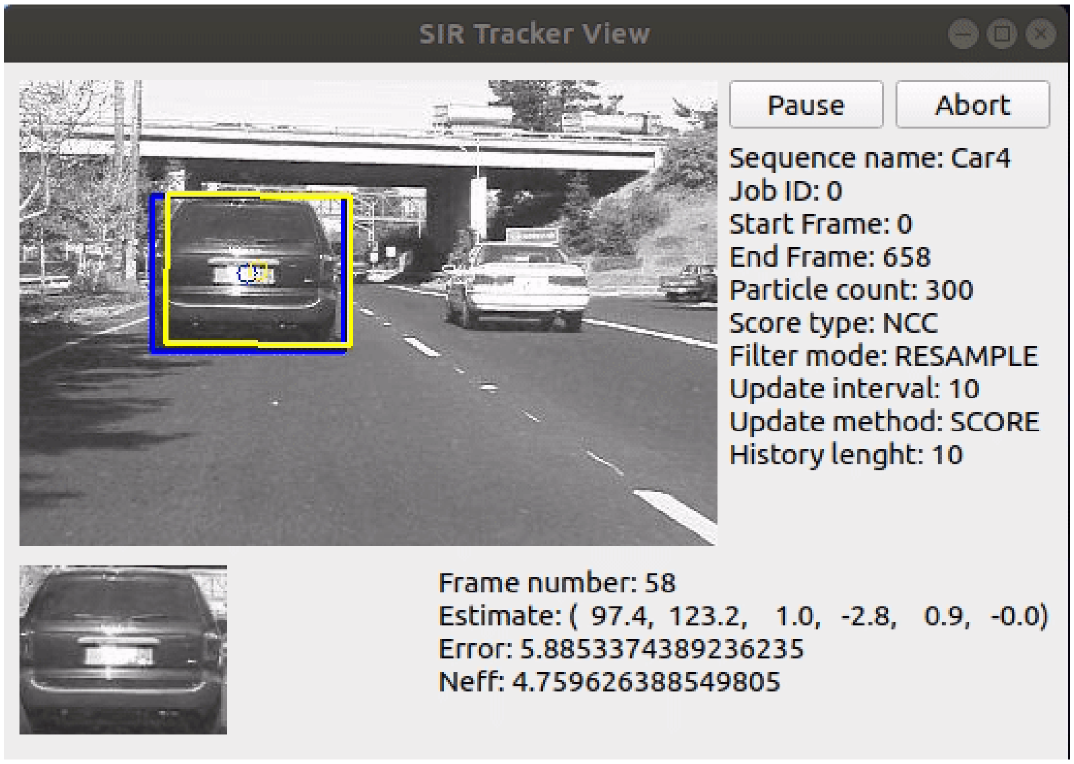
and the main interface can be seen in Figure 3.3. This application loads a JSON configuration file to set up the experimental runs. Qt threads were used to allow simultaneous execution of experimental runs, four at a time in our setup. The SIR Batch Viewer in Figure 3.3 loads a batch of SIR particle filters for processing. Monitoring the progress of runs is seen in the Status column and indicates when the run is generating the Tensorflow graph, running, complete, or skipped in the case it finds existing results. To view the run, a click on the run's row will bring up the SIR Tracker View as seen in Figure 3.4. The main view port upper-left corner and the currently used template is in the lower-left of the plot. Run parameters can be seen in the upper-right. There is a a blue bounding box at ground-truth and a yellow bounding box at the output of the tracker in the main view port and a detailed tracker status in text at the lower right.



Source root: /mnt/data/processedsequences
Save path: /mnt/data/capresults

Name	Run Index	particles	Score Type	Filter Mode	Update Interv	Update Methc	History Lengtl	Status
Car4	0	300	NCC	RESAMPLE	10	SCORE	10	Running
Car4	1	300	NCC	RESAMPLE	10	SCORE	10	Running
Car4	2	300	NCC	RESAMPLE	10	SCORE	10	Running
Car4	3	300	NCC	RESAMPLE	10	SCORE	10	Running
Car4	4	300	NCC	RESAMPLE	10	SCORE	10	Not started
Car4	5	300	NCC	RESAMPLE	10	SCORE	10	Not started
Car4	6	300	NCC	RESAMPLE	10	SCORE	10	Not started
Car4	7	300	NCC	RESAMPLE	10	SCORE	10	Not started
Car4	8	300	NCC	RESAMPLE	10	SCORE	10	Not started
Car4	9	300	NCC	RESAMPLE	10	SCORE	10	Not started
Car4	0	300	NCC	RESAMPLE	10	SCORE	20	Not started
Car4	1	300	NCC	RESAMPLE	10	SCORE	20	Not started
Car4	2	300	NCC	RESAMPLE	10	SCORE	20	Not started
Car4	3	300	NCC	RESAMPLE	10	SCORE	20	Not started
Car4	4	300	NCC	RESAMPLE	10	SCORE	20	Not started
Car4	5	300	NCC	RESAMPLE	10	SCORE	20	Not started
Car4	6	300	NCC	RESAMPLE	10	SCORE	20	Not started
Car4	7	300	NCC	RESAMPLE	10	SCORE	20	Not started
Car4	8	300	NCC	RESAMPLE	10	SCORE	20	Not started
Car4	9	300	NCC	RESAMPLE	10	SCORE	20	Not started
Car4	0	300	NCC	RESAMPLE	10	SCORE	30	Not started
Car4	1	300	NCC	RESAMPLE	10	SCORE	30	Not started
Car4	2	300	NCC	RESAMPLE	10	SCORE	30	Not started
Car4	3	300	NCC	RESAMPLE	10	SCORE	30	Not started
Car4	4	300	NCC	RESAMPLE	10	SCORE	30	Not started
Car4	5	300	NCC	RESAMPLE	10	SCORE	30	Not started

Figure 3.3: SIR Batch Viewer window



Sequence name: Car4
Job ID: 0
Start Frame: 0
End Frame: 658
Particle count: 300
Score type: NCC
Filter mode: RESAMPLE
Update interval: 10
Update method: SCORE
History lenght: 10

Frame number: 58
Estimate: (97.4, 123.2, 1.0, -2.8, 0.9, -0.0)
Error: 5.8853374389236235
Neff: 4.759626388549805

Figure 3.4: Example SIR Tracker View window with Car4 sequence

Chapter 4

Experimental Results

The results presented here show that my proposed methods of expanded likelihood function and SVD template generation can substantially improve particle filter target tracking performance. The improvement to the particle degeneracy problem by means of the proposed 3-frame likelihood function is shown in terms of the effective number of particles N_{eff}. The addition of an SVD template update strategy is shown to provide a more robust method of tracking as demonstrated quantitatively by the resulting track length data. The successful software implementation and collection of the data is evidence that Tensorflow and Python are a great potential platform for current and future investigation into particle filter tracking. Data acquisition was performed by filtering frames so that all collected data are within track tolerance to prevent meaningless values from contaminating the results.

4.1 Improvements by Expanded Likelihood Function

Expanding the likelihood function has the greatest effect on the effective number of particles. While these data do not show that it has any significant impact on the ability to maintain track los for a longer number of frames, a modest improvement in the mean squared error of the tracked centroid is observed at the low particle count of 100 as seen in Table 4.4. This result is significant for real-time deployable systems the ability of a track filter to operate with only a small number of particles is one of the keys to reducing the computational complexity of the particle filtering approach to levels that are feasible for practical commercial and military systems.

4.1.1 Expanded Likelihood Function and Particle Degeneracy

A visual example for the Dudek sequence can be seen in the set of N_{eff} histograms in Figure 4.1 which show a the dramatic improvement to particle counts by auxiliary filtering, as well as incremental improvements by extending the likelihood function across multiple temporal frames.

The resulting composite median N_{eff} calculations over all videos sequences in the experiment can be seen in Table 4.1. Their equivalent as a percentage of particle count is provided in Table 4.2 and in this table it can be seen that the best improvement to effective number of particles of 37% belongs to the proposed extended likelihood function (ASVHO) as indicated in bold-italics. It can be seen from these that the auxiliary particle filtering (ASIR) provides the largest boost to N_{eff} across all groups. Particle degeneracy without ASIR is intensified by the previously mentioned requirement that we use a large covariance in our particle prediction step to accommodate such a diverse set of videos. This large covariance compounds rapidly leading to particle degeneration. As seen in Table 4.2, only 3%-8% of the particles survive which is quite a computational waste when over 90% of the remaining particles are not useful at all in the pdf representation. Without auxiliary filtering the effect on N_{eff} by improved likelihood functions is negligible in our data; however, with auxiliary filtering the effect is quite noticeable.

It can be seen that there is an improvement to the effective number of particles with the 2-frame likelihood function, here labeled ASV consistent with the Augmented State Vector nomenclature established in [4], and the 3-frame likelihood function (ASVHO) shows further improvement still. This improvement was seen across the board for all particle counts in the set as can be seen in Table 4.1. When taken as a percentage of particles, the improvement to N_{eff} can be seen in Table 4.2 to have a similar effect for all particle counts as would be expected intuitively as there is no difference in the life of the particles for different particle counts. The improvement to effective number of particles as a percentage of actual particles with ASV is approximately 4% and the proposed ASVHO across 3 frames has a greater improvement of approximately 10.5%, supporting the original hypothesis that temporally extending the likelihood function would be beneficial to N_{eff}. The N_{eff} results for individual videos vary, but the general theme of improvement remains the same. Median N_{eff} values were calculated individually for each video sequence and are provided in Appendix A.1.

4.1.2 Extended Likelihood Function and Track Length

To investigate the effect of the proposed likelihood function on tracking performance, Table 4.3 hows the average number of frames before track loss, here called track length. In the collected data set it is seen that the greatest benefit occurs due to auxiliary particle filtering and the likelihood function has no discernible impact on track length nor does the number of particles used. This is likely due to the fact that track loss in each video in most cases occurs due to a single event in each sequence, such as entering shadow for the Car4 Sequence,

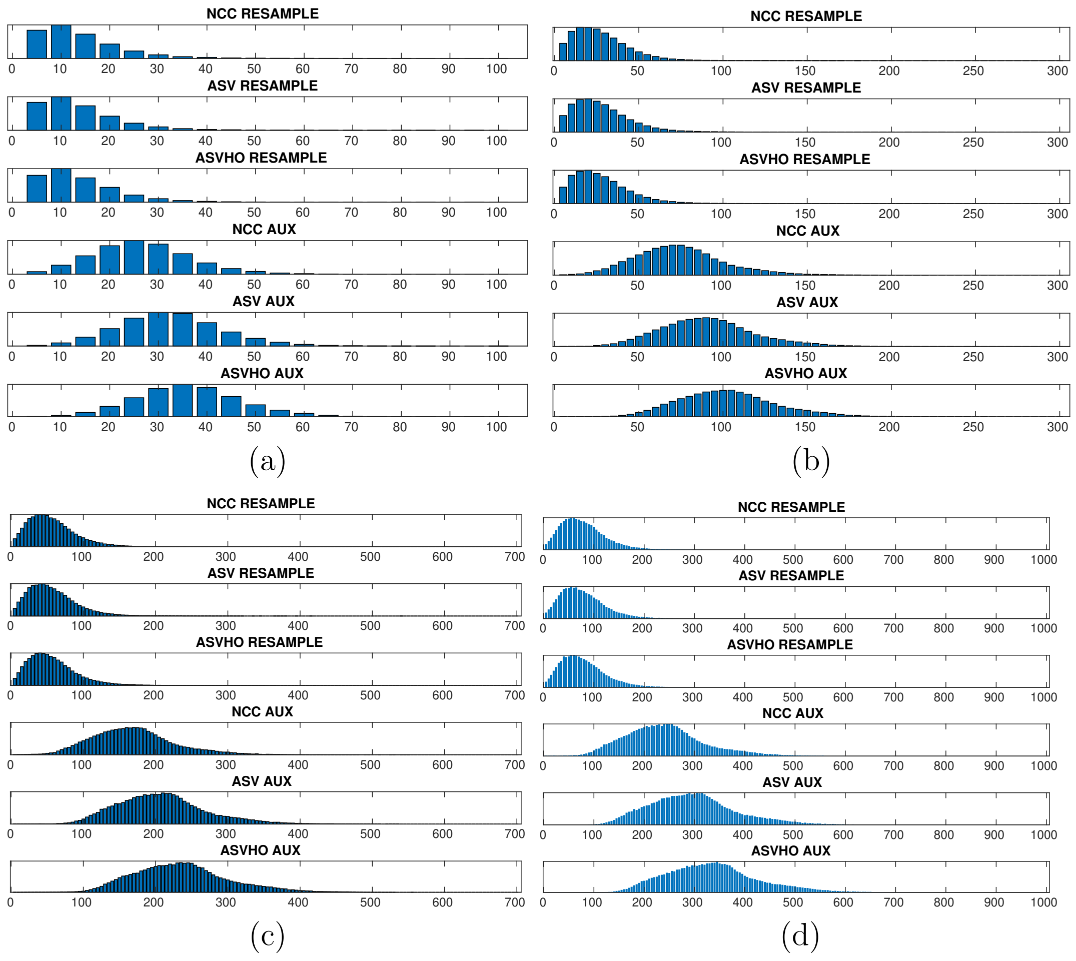


Figure 4.1: Effective number of particles N_{eff} histogram comparison for Dudek at: (a) $N = 100$, (b) $N = 300$, (c) $N = 700$, (d) $N = 1000$

Table 4.1: Composite median N_{eff} over likelihood functions

Particle count	SIR			ASIR		
	NCC	ASV	ASVHO	NCC	ASV	ASVHO
100	7.7	7.7	7.8	26.2	32.9	37.0
300	14.2	14.5	14.4	79.0	97.9	110.1
700	24.5	24.3	24.3	186.0	229.5	257.0
1000	30.4	31.1	30.0	266.5	326.0	364.6

that leads to track loss.

Table 4.2: Composite median N_{eff} over likelihood functions as a percentage of particle count

Particle count	SIR			ASIR		
	NCC	ASV	ASVHO	NCC	ASV	ASVHO
100	7.7%	7.7%	7.8%	26.2%	32.9%	***37.0***%
300	4.7%	4.8%	4.8%	26.3%	32.6%	36.7%
700	3.5%	3.5%	3.5%	26.6%	32.8%	36.7%
1000	3.0%	3.1%	3.0%	26.7%	32.6%	36.5%

Table 4.3: Composite track length over likelihood functions

Particle count	SIR			ASIR		
	NCC	ASV	ASVHO	NCC	ASV	ASVHO
100	78.2	76.8	77.9	221.9	224.5	223.8
300	127.3	124.7	128.6	226.9	228.7	230.9
700	159.8	163.8	160.3	227.1	226.5	228.5
1000	176.9	174.4	175.8	227.4	229.1	226.7

4.1.3 Extended Likelihood Function and Tracking Error

At low particle counts it can be seen in Table 4.4 that there is only a slightly noticeable improvement to tracking error. Above this threshold however the improvement to tracking error is negligible or worsened (at 300 particles) by the ASV and ASVHO. This is likely due to the way templates are extracted by means of the greatest likelihood function in both traditional update and the SVD update strategies. The likelihood function crosses frames and likely introduces error when locally registering the target in the frame from which its ROI is extracted.

4.2 Improvements by SVD Template Generation

SVD generated templates show marked improvement to time before track loss occurs and with one exception show superior tracking error performance. Also

Table 4.4: Composite MSE over likelihood functions

Particle count	SIR			ASIR		
	NCC	ASV	ASVHO	NCC	ASV	ASVHO
100	373.2	378.8	355.2	166.9	159.6	147.9
300	250.2	260.5	252.4	150.2	165.5	153.0
700	224.2	199.5	210.7	168.0	171.0	167.6
1000	207.5	205.9	201.3	174.2	167.6	168.0

observed is a marked improvement to track length and tracking error due to simply extending the depth of the template history. The best performing scores overall in both track length and tracking error belong to the proposed SVD template update method.

4.2.1 SVD Template Generation and Track Length

The best overall track length score of 243.5 frames belongs to the proposed SVD template generation strategy and is indicated by bold-italics in Table 4.5. In Table 4.5 the broad category dividing the table in half shows that Update By SVD improves the time before track loss for every particle count, template update interval, and the size of the length of the historical window of extracted template candidates here called the history depth (h). Increased template history depth at a 10-frame update interval with standard template updating by likelihood function scoring shows improvement across the board due to the availability of better templates from which to select. This effect is not apparent in the longer 20-frame update interval for standard updating by score. Shorter update intervals increase the effect of background leakage and are known to generate poorer performance as can be seen here in Table 4.5. Also expected and apparent from the table is that larger particle counts improve tracking across the board. The most striking performance improvement that can be seen here is the combination of deep template history with the SVD template update. Even with a short 10-frame template update interval, SVD template generation with a deep 30-frame history outperforms the longest 20-frame update interval in the standard template updating by score.

One of the video sequences, rng16_18, in all cases loses track in fewer than ten frames as can be seen in Tables A.5, A.6, A.7, and A.8 in the Appendix. This is due to the fact that the target is very small, only a few pixels in spatial extent, and is nearly indistinguishable from background in the first frame from which the template is extracted. It should be noted that particle filter trackers using a correlation based likelihood function are negatively impacted when the target used for the comparison is only a handful of pixels.

4.2.2 SVD Template Generation and Tracking Error

The track error was calculated using mean-squared-error (MSE) in the tracked target centroid over frames before track loss occurs. The error is in units of pixels squared and lower numbers are better. The best overall tracking error MSE of 104.9 belongs to the SVD template generation update strategy and is indicated by bold-italics in Table 4.6. These values are relatively high due to the use of a template update that has a tendency to move the centroid off center-target from what has been marked in the ground truth. The effect is cumulative until track is fully lost. Often the centroid is marked on different locations of the target as in the CarScale sequence due to the target appearance change due to rotation of the vehicle changing from passenger front facing the camera to passenger rear facing the camera at the end of the sequence. As can be seen in Table 4.6 there is a general trend that longer update intervals

Table 4.5: Composite average time before track loss over template update strategies

Particle count	Update By Score						Update By SVD					
	interval=10			interval=20			interval=10			interval=20		
	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30
100	122.0	127.2	134.4	145.2	136.2	142.9	137.1	160.3	175.4	166.7	175.5	183.0
300	138.0	150.8	156.7	171.7	162.7	171.2	162.0	187.1	213.2	194.9	210.2	215.7
700	149.5	162.5	169.0	192.2	181.3	181.2	168.5	207.1	234.6	216.3	230.7	239.2
1000	154.5	171.3	180.3	198.3	187.5	193.9	176.9	215.1	238.1	223.3	238.0	***243.5***

improve track error. The same reasoning applies here that background leakage compounds faster with shorter update intervals leading to increased error. Update by SVD for shorter 10-frame template histories with short 10-frame update interval shows that error performance is impacted negatively; the tradeoff here is improved track length when comparing error here with track length as in Table 4.5. The remaining combinations show the trend that update by SVD yields superior error performance over the standard template update by best likelihood score when all other parameters are equal.

Table 4.6: Composite MSE Over Template Update Strategies

Particle count	Update By Score						Update By SVD					
	interval=10			interval=20			interval=10			interval=20		
	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30
100	359.8	248.3	203.8	194.2	176.4	193.9	405.0	217.6	150.7	190.5	155.9	129.9
300	338.3	249.9	210.5	187.6	163.3	158.1	364.5	204.1	134.1	135.0	128.5	117.8
700	331.2	239.1	209.6	175.1	172.3	163.9	409.6	200.8	124.2	117.5	113.8	***104.9***
1000	350.4	218.3	196.6	163.3	156.0	151.6	447.4	193.2	125.8	119.8	113.5	105.8

Chapter 5

Conclusion

Improvements to the field of visual tracking to reduce computational cost and provide more robust tracking are highly sought after. Automated visual tracking techniques must accommodate complications such as noisy observations and changing target appearance. In Chapter 2 we touched on several core techniques including image segmentation for identifying potential target regions of interest, target classification and rejection, and template updating. Sequential Bayesian methods were introduced that incorporate known target dynamics to provide for target tracking. Among these techniques, the Monte Carlo method of particle filtering was shown to allow for the tracking of non-linear target dynamics that would otherwise be intractable analytically. The phenomenon of particle degeneracy was discussed as having a detrimental effect, often leading to track loss, on particle filter based target trackers. Auxiliary particle filtering and the likelihood function introduced by the "Augmented State Vector" approach where shown to provide improvement to the particle degeneracy problem thus providing a means for reducing these detrimental effects. The issues of background leakage, dynamic appearance changes, and noisy target observations were identified as key areas that should be a focus for improvement in any proposed template update method due to their impact leading to poor tracking performance and track loss.

In Chapter 3 we proposed a novel SVD based template generation strategy to reduce the detrimental effects of background leakage and to better accommodate for dynamic target appearance changes in order to provide more robust tracking performance. Also proposed in Chapter 3 was an extension to the "Augmented State Vector" approach incorporating 3 temporally adjacent observations to provide better indirect velocity information in the likelihood function thereby providing a noticeable reduction in particle degeneracy.

The experimental results presented in Chapter 4 show a dramatic 25.6% improvement in time before track loss using the best results from the SVD and non-SVD (score) methods, as shown in Table 4.2, indicating significantly more robust performance. The temporal extension of the likelihood function in an auxiliary particle filter proved to be a great benefit in the reduction of particle degeneracy by 11.2%-12.5% when compared to the previous "Augmented State Vector" approach, as shown in Table 4.1. The increased effective number of particles N_{eff} will allow for the use of fewer particles in a PF tracker while maintaining track performance, thereby reducing the requred computational expense. The extended likelihood function was shown to have no significant effect on tracking error. Deeper target observation histories did show a trend towards better performance.

The original contributions of this dissertation include the following:

1. Extended likelihood function incorporating 3 temporally adjacent observations.
2. SVD based target appearance model generation.

3. Implementation using Tensorflow-GPU and Python for GPU coprocessing.

The extended likelihood function provides a noticeable reduction in the particle degeneracy phenomenon. Calculation of the number of effective particles of the “Augmented State Vector” technique was provided here to enhance previous support for use of the method and for use in comparing with the proposed extension. The number of effective particles has a direct effect on the adequacy of the representation of the state pdf for use in particle filter tracking and any increase to it improves state pdf representation. This allows for implementation using fewer real and computationally expensive particles to provide a similar number of effective particles and thus similarly adequate state pdf representation.

SVD based target appearance model generation allows for more robust tracking performance. Building an appearance model by performing SVD from several candidate target regions of interest allows for reduced influence from observation noise and intermittent changes in illumination and target appearance. With the exception of a low template history, the SVD is also shown to improve tracking error as well. Template updating by means of SVD based template generation has proven to provide a more robust update strategy as the increased tracking time has shown.

The use of GPU coprocessing has allowed the collection of the experimental results involving 57,600 runs in a reasonable amount of time to allow for quantitative empirical analysis of the performance benefits that the proposed SVD template update and extended likelihood functions provide. Implementation in Tensorflow has shown that it can be used to perform complex tasks other than its primary use in training of neural networks and provides for a unique GPU accelerated platform for use in further particle filter tracking research.

These proposed methods provide for more robust tracking performance by use of SVD based template generation to reduce the detrimental effects of target appearance change and background leakage. It has been shown that there is improved particle filter adequacy and potential computational cost savings by reduction of the particle degeneracy phenomenon through the proposed enhanced likelihood function.

5.1 Future Work

The alternatives available for use as a likelihood function in a particle filter are boundless and this presents ample opportunity for future work. One alternative came from a misunderstanding of my own while discussing the work in [4] with my colleagues. In attempting the first implementation off-the-cuff as it were I implemented a likelihood function based on the summation of the correlations in temporally adjacent frames and it did show improvement as well although only tested with a single video. The terms of that series could also be weighted in numerous ways such as tapering off over a small window of past frames. The effects of sub-sampled target signatures should also be investigated. It would be worth while to observe the extension of observation across even more temporally adjacent video frames to see where the point of diminishing return lies.

Sub-sampled targets require far less computation when performing correlation calculation and would provide a potential computational cost savings. It would therefore be beneficial to compare the trade off between cost savings due to sub-sampling and the effects it has on the ability to track robustly.

As mentioned in Chapter 2 it is more common that a target acquisition system feeds into target tracking. In this work that job was relegated to the use of ground-truth to seed the initial target location. It would be very interesting to see the performance of the proposed improvements in such a system.

Appendix

Appendix A

Tables

A.1 Median Effective Number of Particles Tables

Table A.1: Effective Number of Particles for $N = 100$

Sequence	SIR			ASIR		
	NCC	ASV	ASVHO	NCC	ASV	ASVHO
Car4	4.4	4.4	4.4	11.3	14.6	17.3
CarScale	4.6	4.0	3.6	27.3	34.8	39.4
Coke	6.1	6.9	6.0	24.3	30.3	36.4
Crossing	5.2	4.5	4.5	25.6	34.8	38.2
Dudek	9.3	9.3	9.4	23.5	29.4	33.3
Football	3.8	3.7	3.7	22.3	28.8	33.0
Girl	4.1	4.6	5.0	27.4	33.1	36.4
RedTeam	4.6	4.6	4.6	37.0	44.8	48.6
Skater	14.8	14.5	14.6	42.4	50.4	54.6
Skater2	10.6	9.9	9.1	24.7	33.1	38.8
bc1 case3	8.2	8.6	8.8	27.0	32.0	35.4
bc3 case7	6.8	7.0	7.1	17.7	21.8	24.9
rng14 15	4.3	4.0	3.8	71.2	74.0	75.6
rng16 18	3.7	3.6	4.6	68.9	75.4	75.3
rng18 16	3.4	2.9	2.7	48.3	56.2	61.8
rng19 06	20.2	29.0	28.0	71.6	79.2	81.8
rng19 13	4.1	5.1	4.7	55.4	62.4	64.7
rng19 NS	3.7	5.3	4.2	52.4	58.8	60.4
walking	3.3	3.1	3.1	13.1	18.6	22.8
woman	4.2	4.3	4.4	20.1	27.6	32.1

Table A.2: Effective Number of Particles for $N = 300$

Sequence	SIR			ASIR		
	NCC	ASV	ASVHO	NCC	ASV	ASVHO
Car4	6.0	6.0	5.8	28.5	38.2	44.5
CarScale	10.2	9.7	10.0	81.4	102.1	114.3
Coke	16.2	14.1	13.9	71.0	90.3	100.0
Crossing	11.1	11.0	11.1	81.9	100.3	112.5
Dudek	22.0	22.4	22.1	69.6	86.5	98.5
Football	6.6	7.0	7.2	65.9	86.1	96.3
Girl	11.2	10.3	11.6	77.5	94.2	104.4
RedTeam	7.6	7.6	7.9	112.0	132.6	143.0
Skater	40.2	38.6	39.2	122.7	144.6	158.0
Skater2	18.0	20.1	20.7	67.5	94.3	108.2
bc1 case3	10.7	10.8	10.6	76.1	91.6	101.4
bc3 case7	7.9	7.9	7.8	50.8	63.1	70.7
rng14 15	7.0	6.1	7.9	206.3	212.1	213.7
rng16 18	7.9	6.3	8.2	198.0	206.9	213.5
rng18 16	5.1	4.0	5.1	149.8	167.3	180.9
rng19 06	64.4	57.0	52.5	218.6	230.2	239.2
rng19 13	7.1	6.9	8.1	164.3	181.4	188.8
rng19 NS	7.2	6.7	8.5	152.5	174.4	178.5
walking	5.2	5.6	5.3	39.0	56.5	67.0
woman	10.2	10.0	10.0	57.1	77.1	91.0

Table A.3: Effective Number of Particles for $N = 700$

Sequence	SIR			ASIR		
	NCC	ASV	ASVHO	NCC	ASV	ASVHO
Car4	8.4	8.5	8.5	61.9	83.1	99.6
CarScale	20.3	20.5	21.0	187.4	233.8	261.6
Coke	28.0	28.2	33.5	156.7	214.0	234.5
Crossing	24.9	24.1	24.5	192.6	238.4	269.0
Dudek	48.7	48.7	48.9	161.9	201.4	228.7
Football	14.5	14.3	14.9	157.1	195.9	219.5
Girl	21.3	22.6	20.2	177.9	216.6	243.0
RedTeam	13.3	13.3	13.5	262.2	307.7	332.8
Skater	86.7	84.8	87.5	278.9	334.7	370.7
Skater2	32.8	32.1	31.3	164.5	211.5	251.9
bc1 case3	17.7	17.6	17.6	175.9	208.4	229.6
bc3 case7	11.4	11.5	11.7	118.5	147.0	164.7
rng14 15	11.1	15.0	12.7	467.7	486.8	487.4
rng16 18	14.2	11.7	10.8	446.1	478.1	482.3
rng18 16	8.8	5.6	5.9	354.2	383.1	412.4
rng19 06	110.7	125.9	119.6	487.6	532.3	554.1
rng19 13	12.9	13.1	13.0	382.3	415.1	432.1
rng19 NS	14.7	11.0	13.0	364.8	403.6	420.5
walking	10.4	10.4	10.5	93.8	132.4	162.3
woman	21.6	20.9	21.4	130.4	176.1	205.1

Table A.4: Effective Number of Particles for $N = 1000$

Sequence	SIR			ASIR		
	NCC	ASV	ASVHO	NCC	ASV	ASVHO
Car4	10.1	10.1	10.0	87.2	116.7	140.5
CarScale	27.7	27.2	26.6	266.1	331.2	370.1
Coke	45.5	42.4	37.2	221.5	298.2	332.6
Crossing	33.1	33.3	33.4	262.2	325.2	367.9
Dudek	69.7	68.7	69.2	232.1	288.3	327.3
Football	19.4	19.9	19.4	223.7	275.5	310.5
Girl	27.6	29.4	27.6	252.4	305.8	334.2
RedTeam	17.2	17.7	17.2	374.5	435.2	472.5
Skater	118.6	119.4	121.2	397.4	480.3	526.7
Skater2	46.9	45.3	43.7	218.9	298.4	324.6
bc1 case3	22.8	22.1	22.1	248.3	296.3	322.8
bc3 case7	13.9	14.0	13.5	170.4	209.3	235.9
rng14 15	17.3	13.1	16.0	658.5	681.3	691.3
rng16 18	18.0	16.6	16.2	653.1	669.6	684.2
rng18 16	9.3	8.1	9.2	482.8	541.2	578.1
rng19 06	168.1	174.0	154.9	710.8	766.8	789.1
rng19 13	17.5	17.0	15.6	545.4	589.8	612.0
rng19 NS	17.4	17.0	15.6	526.3	575.0	603.1
walking	15.2	14.3	14.6	139.7	195.1	232.9
woman	29.1	29.3	27.9	185.8	244.6	293.8

A.2 Track Length Tables

Table A.5: Average track length across template update methods for $N = 100$

Sequence	Update By Score						Update By SVD					
	interval=10			interval=20			interval=10			interval=20		
	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30
Car4	282.9	295.5	321.1	361.9	304.8	328.4	313.4	421.5	441.8	422.2	433.6	425.0
CarScale	142.7	145.0	145.6	148.4	146.0	145.8	151.1	150.6	156.0	154.7	159.2	169.8
Coke	31.5	32.9	33.5	32.1	29.2	30.0	38.1	40.1	31.5	34.9	34.7	35.5
Crossing	36.4	37.7	38.4	41.3	42.4	42.3	35.5	37.9	42.9	42.9	43.3	40.6
Dudek	752.7	719.5	741.0	754.4	746.6	738.9	752.4	830.9	849.4	853.9	873.9	847.2
Football	153.9	136.9	122.5	130.4	115.8	120.5	176.1	189.0	170.3	158.5	140.6	163.9
Girl	84.8	85.4	78.0	104.0	92.9	89.1	108.8	102.8	100.4	113.9	118.6	122.4
RedTeam	418.9	502.0	571.0	575.4	561.8	639.0	436.8	595.4	837.5	692.5	808.2	933.4
Skater	124.1	119.2	110.1	127.9	114.6	125.0	124.8	130.0	134.8	141.1	143.3	152.7
Skater2	46.3	49.1	53.0	55.1	43.8	43.8	42.5	49.8	59.9	61.0	56.0	61.4
bc1 case3	74.8	91.1	117.8	136.2	130.2	134.7	74.5	102.6	142.2	141.7	140.3	154.7
bc3 case7	103.1	119.2	121.4	126.8	119.4	123.1	119.9	134.9	135.3	141.9	135.8	137.1
rng14 15	26.4	31.9	36.6	48.3	39.8	39.6	53.9	61.4	66.9	60.0	62.0	67.1
rng16 18	8.2	8.2	8.2	8.1	8.1	8.1	8.1	8.1	8.1	8.1	8.1	8.1
rng18 16	17.2	19.8	28.5	26.6	28.1	30.1	32.5	48.3	39.2	28.0	47.9	40.6
rng19 06	2.9	2.9	2.9	2.9	2.9	2.9	2.9	2.9	2.9	2.9	2.9	2.9
rng19 13	22.9	27.6	27.4	55.8	38.7	43.3	83.0	81.5	82.1	79.7	76.9	81.6
rng19 NS	14.2	14.2	14.0	28.8	19.9	19.9	28.8	28.7	28.9	29.2	29.4	29.1
walking	30.6	36.9	42.8	54.9	55.8	70.7	35.9	48.1	65.1	60.9	62.2	74.4
woman	65.1	68.7	75.1	85.6	82.7	81.9	123.9	141.3	113.0	106.4	134.0	113.3

Table A.6: Average track length across template update methods for $N = 300$

Sequence	Update By Score						Update By SVD					
	interval=10			interval=20			interval=10			interval=20		
	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30
Car4	310.8	332.8	345.2	428.4	416.9	383.7	349.1	498.7	527.0	504.7	516.3	495.4
CarScale	204.8	202.2	194.5	207.7	207.3	189.5	212.9	213.6	204.8	208.6	210.9	202.5
Coke	35.6	36.9	36.6	33.4	33.3	33.8	40.6	44.9	31.2	36.1	36.7	36.1
Crossing	39.2	38.3	39.3	43.9	45.3	44.1	42.6	40.4	45.5	46.3	54.7	49.4
Dudek	809.3	862.7	905.4	908.9	842.2	920.8	818.1	930.5	998.0	997.0	989.8	996.0
Football	191.5	196.1	156.9	177.0	158.9	164.6	227.0	228.7	238.3	221.9	194.8	209.5
Girl	110.6	108.2	100.0	129.8	121.3	124.7	118.8	136.6	140.5	154.3	157.2	166.7
RedTeam	448.4	549.7	651.1	659.4	617.4	721.3	608.2	726.5	1050.1	779.4	1058.1	1134.7
Skater	140.3	133.9	127.0	138.6	130.6	131.0	138.2	133.9	143.2	149.4	149.1	156.2
Skater2	50.8	65.1	61.3	80.0	69.7	67.6	56.0	59.9	77.9	77.7	58.6	78.3
bc1 case3	83.2	112.6	137.1	149.0	154.5	160.6	84.9	112.4	158.6	154.9	154.8	170.0
bc3 case7	121.3	140.6	141.0	141.4	143.6	143.2	135.1	145.1	155.2	151.3	152.1	152.0
rng14 15	27.0	32.2	33.4	50.7	41.9	49.1	54.1	64.4	80.0	68.5	72.9	83.4
rng16 18	8.7	8.7	8.7	8.7	8.7	8.7	8.7	8.7	8.7	8.7	8.7	8.7
rng18 16	21.9	20.5	20.5	32.0	27.5	30.5	23.0	74.6	70.9	27.6	58.1	39.1
rng19 06	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
rng19 13	28.0	27.6	26.7	55.5	42.5	45.4	92.8	92.9	89.4	87.6	89.2	88.2
rng19 NS	16.5	17.2	17.1	30.0	25.5	25.2	31.1	30.4	30.9	30.5	30.4	30.7
walking	34.2	38.7	43.5	64.0	64.2	73.5	34.5	50.4	77.9	67.9	70.8	86.8
woman	74.8	88.3	86.5	92.2	100.9	104.5	161.0	147.3	133.8	121.9	138.3	127.1

Table A.7: Average track length across template update methods for $N = 700$

Sequence	Update By Score						Update By SVD					
	interval=10			interval=20			interval=10			interval=20		
	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30
Car4	310.2	382.4	391.4	487.3	467.2	400.3	338.4	575.7	601.3	594.2	582.8	591.2
CarScale	220.4	228.2	233.9	239.1	247.0	242.4	243.7	243.5	243.4	247.0	233.8	243.4
Coke	34.6	36.9	36.8	33.6	31.0	31.2	41.0	42.7	28.2	41.9	36.1	32.5
Crossing	40.2	41.4	43.2	43.0	43.4	43.4	46.5	43.1	45.7	45.9	65.5	42.7
Dudek	780.7	864.5	932.2	965.3	954.3	905.9	767.1	912.9	998.0	998.0	998.0	998.0
Football	219.2	197.8	180.6	204.0	175.8	178.0	273.2	269.9	269.9	243.6	219.9	256.1
Girl	120.5	119.9	105.5	162.7	105.0	114.0	139.5	155.0	174.0	162.5	182.7	204.8
RedTeam	598.9	677.5	703.7	794.1	742.0	823.6	685.2	928.8	1240.2	970.6	1235.1	1315.3
Skater	146.3	127.0	129.2	145.3	130.1	133.1	136.3	142.6	145.3	152.2	147.0	159.0
Skater2	62.3	63.9	58.1	87.8	70.1	66.2	33.5	51.8	70.5	64.9	57.1	78.2
bc1 case3	87.7	116.1	147.2	164.5	161.0	171.7	88.8	119.5	174.2	174.8	173.1	178.1
bc3 case7	131.0	146.9	151.4	155.8	155.3	156.0	134.2	155.3	163.6	162.1	161.5	164.6
rng14 15	29.9	37.2	37.3	51.5	43.0	44.1	55.6	64.1	78.2	81.0	76.1	103.5
rng16 18	9.6	9.6	9.6	9.6	9.6	9.6	9.6	9.6	9.6	9.6	9.6	9.6
rng18 16	23.2	21.1	23.7	30.9	36.4	30.4	28.7	73.2	78.5	23.9	50.4	39.8
rng19 06	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
rng19 13	28.8	29.0	29.3	69.2	41.6	43.7	105.5	97.3	97.2	104.3	98.2	99.0
rng19 NS	23.8	24.6	24.9	32.9	30.7	30.7	33.3	33.3	33.7	34.0	34.6	34.8
walking	35.2	43.8	50.5	69.3	72.8	87.0	37.0	51.2	83.8	77.3	79.7	99.0
woman	84.8	79.5	87.8	95.3	107.1	110.3	169.8	169.9	153.1	134.8	169.6	131.4

Table A.8: Average track length across template update methods for $N = 1000$

Sequence	Update By Score						Update By SVD					
	interval=10			interval=20			interval=10			interval=20		
	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30
Car4	298.0	422.5	399.8	489.1	470.3	481.3	315.9	614.1	637.1	597.9	621.3	569.4
CarScale	226.0	228.1	234.1	245.2	242.8	247.1	247.1	247.1	244.5	247.1	243.5	247.1
Coke	35.0	36.7	36.0	32.4	31.6	31.8	39.0	47.6	28.9	43.6	37.9	32.4
Crossing	45.6	46.3	45.7	48.9	47.2	46.7	46.7	46.9	44.5	45.1	64.7	45.5
Dudek	767.6	828.0	931.9	976.0	935.7	927.9	805.0	950.6	998.0	998.0	998.0	998.0
Football	229.2	201.9	184.2	208.5	189.5	171.2	284.1	279.8	278.0	260.6	196.6	254.6
Girl	115.9	117.3	120.0	159.2	116.2	122.7	134.2	163.2	173.2	173.4	182.1	197.7
RedTeam	705.5	793.5	875.1	854.0	859.9	961.0	781.5	936.9	1227.3	1016.1	1317.0	1394.5
Skater	145.1	132.8	130.4	146.5	119.1	124.1	150.5	146.7	140.3	157.2	153.9	159.0
Skater2	70.2	68.0	66.9	103.4	80.0	85.6	34.4	58.4	73.6	80.7	64.3	81.2
bc1 case3	88.1	117.8	148.8	172.6	161.6	167.5	89.8	124.7	177.2	176.9	176.1	178.3
bc3 case7	132.9	150.4	156.9	159.3	160.2	162.6	141.5	154.4	166.2	164.6	167.0	166.5
rng14 15	33.6	36.8	45.2	57.4	45.3	42.5	60.1	72.1	108.1	89.7	92.8	113.6
rng16 18	9.7	9.7	9.7	9.7	9.7	9.7	9.7	9.7	9.7	9.7	9.7	9.7
rng18 16	24.2	27.9	26.6	35.4	33.3	31.3	26.0	83.0	66.8	26.1	39.7	36.8
rng19 06	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
rng19 13	22.3	20.5	25.0	71.5	35.7	40.9	109.0	106.7	104.3	104.7	104.9	102.3
rng19 NS	21.9	22.2	22.3	36.5	30.6	30.6	36.8	37.4	38.0	37.5	38.1	37.6
walking	34.8	43.3	49.4	67.7	68.9	82.5	37.1	53.1	84.0	77.5	78.3	100.0
woman	82.0	118.5	95.0	89.7	108.8	108.7	187.0	167.4	159.6	157.4	170.5	142.0

Table A.9: Average track length across likelihood functions for $N = 100$

Sequence	SIR			ASIR		
	NCC	ASV	ASVHO	NCC	ASV	ASVHO
Car4	158.7	155.0	184.1	548.6	568.9	560.8
CarScale	62.6	53.3	40.5	251.0	250.3	249.7
Coke	28.9	28.3	30.9	39.8	35.7	38.4
Crossing	31.8	30.0	31.1	47.9	52.4	47.5
Dudek	656.8	633.1	659.4	917.8	930.4	932.8
Football	66.5	67.7	43.4	238.6	242.0	231.0
Girl	28.9	33.9	32.3	169.6	163.6	172.2
RedTeam	58.7	56.5	50.9	1203.4	1209.2	1207.3
Skater	118.4	116.5	116.5	141.3	139.2	141.9
Skater2	42.7	39.5	56.3	48.6	65.1	58.7
bc1 case3	89.7	94.2	91.8	149.5	147.8	147.5
bc3 case7	88.6	94.5	90.7	163.1	160.3	161.8
rng14 15	9.1	6.8	8.2	91.4	99.6	81.8
rng16 18	5.1	7.1	6.6	10.0	10.0	10.0
rng18 16	8.1	6.9	7.7	62.4	55.2	53.1
rng19 06	2.9	2.8	2.5	3.0	3.0	3.0
rng19 13	12.2	11.2	10.2	103.5	104.2	108.9
rng19 NS	8.3	9.9	9.1	38.4	36.7	40.2
walking	42.0	41.9	38.8	65.3	66.5	64.7
woman	43.5	47.0	46.2	144.6	149.6	164.7

Table A.10: Average track length across likelihood functions for $N = 300$

Sequence	SIR			ASIR		
	NCC	ASV	ASVHO	NCC	ASV	ASVHO
Car4	304.1	257.0	295.9	561.6	569.1	566.7
CarScale	159.7	151.3	170.1	250.3	250.3	247.8
Coke	33.0	30.8	32.1	43.5	40.3	37.9
Crossing	44.7	39.9	35.9	47.5	48.0	48.4
Dudek	887.3	885.9	892.2	912.0	954.2	957.6
Football	140.9	146.4	152.8	243.6	247.8	251.1
Girl	98.1	91.9	95.9	165.4	165.3	167.8
RedTeam	241.0	240.6	249.2	1236.8	1252.7	1281.9
Skater	129.6	130.4	135.9	146.4	145.9	147.7
Skater2	69.6	67.3	72.8	66.5	64.7	60.6
bc1 case3	117.9	122.8	118.0	152.5	150.7	154.4
bc3 case7	123.2	117.9	124.1	164.7	165.7	165.4
rng14 15	14.2	15.3	12.0	89.4	97.8	100.2
rng16 18	8.3	6.6	7.4	10.0	10.0	10.0
rng18 16	8.8	10.1	7.5	73.1	58.0	65.5
rng19 06	3.0	2.9	3.0	3.0	3.0	3.0
rng19 13	19.8	25.8	23.1	115.9	96.7	101.6
rng19 NS	9.1	13.6	11.7	43.1	39.2	41.0
walking	52.3	51.3	54.4	65.7	63.7	65.8
woman	82.1	85.1	78.4	146.5	151.8	144.3

Table A.11: Average track length across likelihood functions for $N = 700$

Sequence	SIR			ASIR		
	NCC	ASV	ASVHO	NCC	ASV	ASVHO
Car4	398.3	397.8	395.7	556.8	558.6	554.0
CarScale	236.6	215.8	234.1	247.7	250.3	248.3
Coke	34.0	32.0	32.9	38.6	38.3	37.4
Crossing	43.3	41.1	46.3	41.5	50.1	49.7
Dudek	914.7	947.6	896.0	919.5	909.2	950.4
Football	198.6	185.9	211.4	255.8	248.8	243.5
Girl	128.1	140.6	107.9	158.8	171.1	166.6
RedTeam	486.9	580.3	553.3	1248.5	1247.4	1241.1
Skater	135.2	136.5	135.8	141.1	151.9	146.3
Skater2	66.7	52.8	54.3	71.0	69.8	67.4
bc1 case3	138.1	140.5	138.7	153.5	153.2	154.2
bc3 case7	141.7	140.8	139.8	164.8	165.9	165.9
rng14 15	20.0	14.7	16.7	102.4	90.0	106.8
rng16 18	9.1	9.2	9.1	10.0	10.0	10.0
rng18 16	13.6	10.3	9.3	69.6	65.7	61.6
rng19 06	3.0	3.0	3.0	3.0	3.0	3.0
rng19 13	36.8	40.6	38.6	106.2	97.3	102.2
rng19 NS	23.8	14.3	13.7	43.9	43.0	47.0
walking	63.6	64.0	61.1	67.5	69.3	67.8
woman	103.5	107.5	109.1	141.2	137.9	147.6

Table A.12: Average track length across likelihood functions for $N = 1000$

Sequence	SIR			ASIR		
	NCC	ASV	ASVHO	NCC	ASV	ASVHO
Car4	446.3	424.5	429.0	538.7	565.9	554.0
CarScale	242.0	224.3	238.3	248.3	249.7	247.3
Coke	33.5	31.1	34.5	42.0	36.7	38.6
Crossing	45.6	42.5	41.9	50.4	53.4	53.1
Dudek	910.8	927.9	911.3	934.8	937.5	935.1
Football	218.6	205.9	212.2	244.4	244.6	243.3
Girl	126.9	140.8	131.9	153.2	168.2	166.5
RedTeam	717.8	701.7	714.1	1258.6	1248.6	1220.2
Skater	139.6	142.6	138.0	142.9	146.7	143.1
Skater2	62.9	68.8	64.0	73.2	83.2	81.2
bc1 case3	142.4	141.1	144.2	154.9	151.7	155.4
bc3 case7	146.2	148.2	149.9	165.2	166.0	165.7
rng14 15	25.4	20.2	27.9	109.2	110.3	105.8
rng16 18	9.0	10.0	8.9	10.0	10.0	10.0
rng18 16	11.2	13.3	11.8	65.5	62.0	64.9
rng19 06	3.0	3.0	3.0	3.0	3.0	3.0
rng19 13	46.5	41.3	45.8	97.8	94.1	98.3
rng19 NS	30.8	16.8	20.2	40.5	43.5	42.9
walking	62.2	63.0	62.2	67.0	68.7	65.3
woman	118.2	120.8	126.2	148.7	138.1	141.3

A.3 Error Tables

Table A.13: MSE Over Template Update Strategies for $N = 100$

Sequence	Update By Score						Update By SVD					
	interval=10			interval=20			interval=10			interval=20		
	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30
Car4	161.9	108.4	87.0	82.7	99.1	48.9	142.6	56.5	33.7	37.6	29.8	22.0
CarScale	534.8	214.9	199.1	114.0	166.5	236.4	395.8	148.2	191.4	147.6	256.9	262.5
Coke	159.6	150.8	146.6	113.1	121.4	121.0	109.9	150.9	122.4	124.1	106.4	118.0
Crossing	32.3	30.1	27.5	15.7	19.1	17.8	28.7	21.7	17.5	12.3	15.2	16.4
Dudek	839.8	614.2	495.7	476.3	407.8	511.8	1169.2	607.6	399.5	509.1	394.7	338.2
Football	73.4	61.9	55.5	48.3	44.2	50.6	71.5	59.8	58.1	44.9	46.4	65.5
Girl	59.7	60.6	54.1	77.9	64.5	52.7	68.7	91.5	87.9	92.4	93.9	74.5
RedTeam	26.8	26.5	21.3	22.7	23.3	19.9	10.5	18.1	20.9	20.6	20.3	18.0
Skater	317.3	327.9	372.5	406.1	383.3	376.7	266.1	263.4	285.1	316.6	258.8	178.0
Skater2	744.2	809.9	759.1	1023.6	836.3	845.9	898.1	981.7	820.2	1001.3	918.5	821.2
bc1 case3	42.0	25.6	18.8	9.8	10.7	8.5	46.8	18.8	11.3	9.2	10.9	6.5
bc3 case7	31.2	20.0	16.5	11.5	11.4	9.7	28.4	15.2	10.6	10.0	11.3	8.7
rng14 15	3.5	3.3	3.4	7.6	4.5	3.6	4.0	3.5	4.1	12.1	3.9	4.2
rng16 18	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
rng18 16	3.2	3.4	4.2	3.4	4.0	4.5	4.2	6.0	3.7	3.6	5.5	3.8
rng19 06	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
rng19 13	4.9	4.3	4.4	6.4	4.7	4.4	5.0	4.1	3.8	4.7	4.0	3.9
rng19 NS	4.5	4.4	4.3	4.6	3.7	3.5	2.4	2.3	2.5	2.7	2.7	2.6
walking	67.1	67.6	67.4	58.7	55.5	54.1	73.1	63.4	54.2	62.5	64.4	50.6
woman	64.9	65.5	61.9	120.5	58.1	65.7	102.7	112.6	74.3	99.0	97.6	80.1

Table A.14: MSE Over Template Update Strategies for $N = 300$

Sequence	Update By Score						Update By SVD					
	interval=10			interval=20			interval=10			interval=20		
	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30
Car4	121.9	71.6	52.9	51.7	68.6	49.6	119.7	34.8	20.5	33.0	20.8	12.0
CarScale	496.3	274.2	331.4	208.2	286.0	230.9	377.0	136.1	188.8	135.3	199.0	288.6
Coke	146.0	147.1	133.6	113.7	128.9	122.7	101.7	162.2	152.3	117.4	101.1	125.5
Crossing	28.4	28.8	27.8	12.4	14.3	13.8	35.5	20.1	16.0	15.0	13.5	12.7
Dudek	807.4	603.2	456.8	446.9	343.8	354.8	1099.9	600.1	357.8	306.1	321.8	291.8
Football	89.5	75.4	60.0	49.9	51.4	55.6	77.5	55.7	58.8	45.9	54.9	69.4
Girl	61.7	61.1	56.0	83.2	61.1	63.3	57.5	69.8	81.4	78.9	96.5	88.5
RedTeam	23.9	22.6	22.3	21.6	24.7	21.7	11.5	17.6	20.2	20.3	20.4	16.2
Skater	278.9	351.2	491.0	378.9	446.2	435.8	260.2	283.5	261.4	289.6	347.1	150.7
Skater2	905.1	769.9	866.6	855.3	634.9	635.2	945.8	821.2	697.2	927.0	730.6	742.7
bc1 case3	41.4	21.3	15.9	7.3	9.0	6.9	40.8	16.0	8.2	7.6	8.7	5.1
bc3 case7	25.2	15.3	12.0	7.0	8.6	7.0	25.7	12.4	7.6	6.8	7.3	5.3
rng14 15	3.0	3.4	3.3	6.9	3.3	7.1	3.7	3.5	5.6	11.2	4.3	5.3
rng16 18	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
rng18 16	3.2	3.1	3.1	3.6	3.0	4.5	3.2	6.4	5.5	3.5	5.2	3.7
rng19 06	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
rng19 13	4.6	5.1	4.6	5.1	4.7	4.4	5.4	4.5	4.0	4.8	4.3	3.7
rng19 NS	4.4	5.6	5.0	4.3	4.2	4.0	2.3	2.3	2.5	2.3	2.3	2.3
walking	62.2	59.8	60.0	56.3	54.2	52.1	67.5	55.0	56.5	55.8	52.5	49.8
woman	51.6	85.5	46.0	65.0	90.5	72.0	136.6	139.0	98.3	80.3	108.6	90.4

Table A.15: MSE Over Template Update Strategies for $N = 700$

Sequence	Update By Score						Update By SVD					
	interval=10			interval=20			interval=10			interval=20		
	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30
Car4	82.4	61.7	62.2	52.5	68.6	31.4	124.5	39.1	17.1	23.3	14.3	10.7
CarScale	532.4	332.3	252.7	138.7	219.6	302.6	423.7	142.3	133.2	140.7	185.3	204.7
Coke	150.5	157.8	156.0	165.4	132.8	132.6	101.8	147.1	142.1	121.5	91.4	101.5
Crossing	24.6	22.7	21.4	12.8	13.1	12.9	33.5	21.3	16.6	9.8	12.1	12.5
Dudek	898.7	605.0	482.8	470.8	377.8	357.2	1408.9	664.8	346.4	294.0	299.4	281.5
Football	81.5	63.0	64.3	53.4	51.5	47.5	74.9	58.9	63.3	47.2	62.2	80.3
Girl	56.6	58.3	55.0	79.7	51.6	46.1	59.5	84.9	107.6	95.8	105.7	115.2
RedTeam	23.0	23.5	24.8	22.6	24.5	20.0	10.1	15.3	18.7	18.0	18.7	14.1
Skater	267.4	393.6	570.7	296.8	589.5	665.3	354.3	257.9	263.1	289.3	277.3	156.9
Skater2	765.6	780.5	804.4	650.5	710.7	676.3	967.3	860.9	886.2	804.6	716.1	761.7
bc1 case3	40.0	20.2	14.2	6.6	8.4	6.5	39.1	15.3	8.0	6.9	8.1	4.4
bc3 case7	25.7	15.0	11.5	6.6	7.1	5.9	25.3	11.7	7.0	5.9	6.6	4.7
rng14 15	3.5	4.5	4.5	8.7	3.9	3.6	2.9	2.9	4.6	14.4	4.1	5.9
rng16 18	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
rng18 16	3.8	3.2	3.6	3.1	5.6	3.4	3.5	6.5	6.2	3.4	4.5	4.2
rng19 06	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
rng19 13	5.1	4.2	4.8	4.7	3.9	3.6	5.2	4.5	3.9	4.5	4.2	3.6
rng19 NS	6.0	7.4	6.4	4.9	5.7	5.6	2.5	2.4	2.5	2.8	2.5	2.5
walking	60.1	59.5	61.0	47.8	50.8	46.7	59.2	53.1	53.8	51.2	52.4	44.7
woman	51.1	49.2	52.7	50.9	65.7	81.9	122.7	166.3	139.3	87.4	145.9	48.2

Table A.16: MSE Over Template Update Strategies for $N = 1000$

Sequence	Update By Score						Update By SVD					
	interval=10			interval=20			interval=10			interval=20		
	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30	h=10	h=20	h=30
Car4	86.0	75.1	50.4	30.4	76.2	30.4	100.4	32.7	18.1	20.1	14.1	9.5
CarScale	591.3	188.7	230.2	134.8	130.8	212.6	423.7	132.3	153.1	138.8	146.2	209.7
Coke	141.5	144.8	134.0	137.0	137.5	136.6	97.6	182.6	149.2	148.9	119.3	126.9
Crossing	30.8	30.6	30.3	9.8	9.8	9.7	33.6	23.6	17.4	9.3	11.3	11.5
Dudek	992.3	591.3	466.0	424.0	366.3	357.9	1574.8	624.2	344.0	296.9	295.5	280.3
Football	84.1	71.9	62.2	53.7	55.7	44.4	72.7	57.4	64.8	50.9	61.4	81.1
Girl	62.0	52.4	51.4	88.2	51.9	48.7	55.5	94.2	102.0	100.6	112.6	104.7
RedTeam	33.1	30.6	23.9	25.6	21.8	21.5	10.2	15.3	17.9	17.9	18.7	14.2
Skater	297.7	415.4	646.1	289.2	529.9	569.8	389.2	249.3	359.0	287.5	284.3	143.9
Skater2	680.4	803.5	854.4	778.0	724.0	777.0	1101.4	942.4	825.8	802.9	972.0	862.1
bc1 case3	41.3	21.5	16.4	6.5	8.3	6.5	39.3	15.4	7.7	6.5	7.9	4.3
bc3 case7	25.4	13.0	10.2	5.8	6.5	5.5	24.7	11.4	6.8	5.7	6.2	4.6
rng14 15	3.1	3.1	5.0	8.6	4.4	3.0	3.0	2.9	6.2	14.6	5.1	5.9
rng16 18	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
rng18 16	3.5	4.5	3.8	3.4	3.6	2.9	3.1	7.9	6.4	3.3	4.7	3.0
rng19 06	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
rng19 13	3.6	3.1	3.4	5.1	3.5	3.9	5.2	4.2	3.6	4.3	4.0	3.4
rng19 NS	5.7	6.4	6.2	4.0	4.1	4.0	2.2	2.1	2.1	2.3	2.1	2.4
walking	61.5	58.6	61.0	49.1	48.8	46.2	57.3	55.0	54.2	50.4	51.3	45.4
woman	55.1	115.8	71.0	46.1	63.1	53.7	161.5	175.8	150.4	117.4	143.9	84.5

Table A.17: MSE over likelihood functions for $N = 100$

Sequence	SIR			ASIR		
	NCC	ASV	ASVHO	NCC	ASV	ASVHO
Car4	123.3	134.1	102.4	54.9	57.7	53.4
CarScale	378.6	397.5	150.0	227.6	229.9	206.4
Coke	151.1	119.9	151.1	125.3	114.9	114.2
Crossing	22.2	22.7	24.7	20.2	18.3	19.4
Dudek	671.4	684.9	622.6	526.6	492.1	440.6
Football	40.6	36.5	55.4	61.6	60.9	60.3
Girl	44.5	48.6	42.4	79.6	79.6	81.0
RedTeam	19.9	21.3	20.2	20.1	20.9	20.9
Skater	274.4	321.3	305.0	327.7	317.6	300.2
Skater2	996.5	1126.9	997.4	868.9	661.1	742.4
bc1 case3	20.2	20.2	22.1	11.7	12.7	12.5
bc3 case7	24.8	22.8	21.5	10.0	10.8	10.2
rng14 15	2.0	1.2	2.0	4.5	7.0	4.0
rng16 18	1.9	2.1	1.3	0.5	0.5	0.5
rng18 16	3.7	3.1	2.6	5.1	4.2	4.0
rng19 06	1.3	1.0	1.1	1.2	0.9	1.1
rng19 13	6.7	5.6	7.7	4.3	4.3	4.1
rng19 NS	9.3	3.5	7.5	2.5	2.4	2.6
walking	73.5	64.9	95.1	50.2	49.7	49.3
woman	30.9	32.7	34.4	106.0	100.0	103.5

Table A.18: MSE over likelihood functions for $N = 300$

Sequence	SIR			ASIR		
	NCC	ASV	ASVHO	NCC	ASV	ASVHO
Car4	79.3	55.9	52.4	46.5	41.4	40.5
CarScale	367.2	316.5	308.5	217.2	237.4	200.7
Coke	100.1	115.4	129.7	157.3	133.0	130.4
Crossing	19.1	22.8	14.2	18.8	21.4	18.9
Dudek	480.8	523.5	497.0	449.2	509.7	459.2
Football	51.2	54.3	60.2	62.4	66.0	70.4
Girl	49.5	46.5	52.4	83.5	89.5	87.6
RedTeam	16.1	16.7	14.7	20.3	21.1	20.7
Skater	302.1	315.5	285.9	324.3	353.2	372.2
Skater2	822.2	704.7	875.3	817.4	694.1	818.8
bc1 case3	14.8	14.8	14.6	11.8	12.2	12.1
bc3 case7	14.9	12.9	12.8	9.8	9.7	9.6
rng14 15	1.8	1.7	1.5	5.1	6.6	5.7
rng16 18	1.7	1.9	1.4	0.4	0.4	0.4
rng18 16	2.8	2.7	2.8	5.0	4.5	4.6
rng19 06	0.9	1.2	1.6	0.8	0.8	0.8
rng19 13	5.4	5.7	5.6	4.3	4.5	4.2
rng19 NS	4.8	6.6	4.5	2.5	2.6	3.0
walking	56.6	67.3	58.4	51.8	52.0	50.8
woman	31.4	47.6	44.4	132.5	124.8	114.3

Table A.19: MSE over likelihood functions for $N = 700$

Sequence	SIR			ASIR		
	NCC	ASV	ASVHO	NCC	ASV	ASVHO
Car4	46.9	49.1	41.7	48.5	34.6	40.6
CarScale	279.7	292.7	281.0	200.8	225.8	218.4
Coke	121.0	123.9	121.9	142.9	155.4	127.2
Crossing	18.2	15.8	18.6	16.5	18.7	17.2
Dudek	545.8	465.3	513.4	527.6	533.0	515.5
Football	63.5	57.0	61.0	65.4	65.4	64.4
Girl	64.2	65.3	62.1	79.9	99.0	103.3
RedTeam	17.1	18.2	15.5	19.8	19.7	20.0
Skater	339.8	314.7	335.1	347.5	436.8	357.5
Skater2	581.7	707.2	733.8	891.2	832.3	837.1
bc1 case3	13.1	12.9	12.9	11.5	11.8	11.8
bc3 case7	12.0	11.4	11.2	9.7	10.0	9.8
rng14 15	1.5	1.3	1.7	6.3	6.1	6.8
rng16 18	0.9	1.0	1.7	0.4	0.4	0.4
rng18 16	2.8	2.8	2.6	5.0	4.9	5.3
rng19 06	0.9	0.9	0.8	0.8	0.7	0.7
rng19 13	3.9	4.4	3.7	4.4	4.5	4.5
rng19 NS	5.1	4.6	8.1	3.0	3.1	4.1
walking	50.7	52.6	51.0	52.9	52.8	51.5
woman	54.3	58.6	58.8	141.1	119.1	127.6

Table A.20: MSE over likelihood functions for $N = 1000$

Sequence	SIR			ASIR		
	NCC	ASV	ASVHO	NCC	ASV	ASVHO
Car4	47.1	42.7	52.5	34.0	35.3	29.3
CarScale	230.3	249.2	229.7	214.4	217.7	195.1
Coke	128.6	129.2	135.8	154.4	149.8	131.4
Crossing	19.2	18.4	17.6	18.9	19.5	18.3
Dudek	551.4	527.7	515.5	555.1	499.5	508.9
Football	64.9	60.9	61.5	65.2	66.5	64.3
Girl	64.7	63.0	68.5	78.4	99.5	105.0
RedTeam	19.0	20.4	20.7	19.9	20.7	20.2
Skater	341.0	340.2	332.6	342.6	402.4	405.9
Skater2	797.8	686.7	828.1	858.8	885.8	870.7
bc1 case3	13.1	13.3	13.2	11.5	12.3	12.0
bc3 case7	10.7	10.6	10.7	9.4	9.5	9.7
rng14 15	1.7	1.5	1.6	7.3	6.3	7.2
rng16 18	0.9	1.1	1.1	0.4	0.4	0.4
rng18 16	2.2	3.2	2.5	5.3	5.0	5.2
rng19 06	0.9	0.7	1.2	0.7	0.7	0.7
rng19 13	3.2	3.4	2.9	4.5	4.6	4.5
rng19 NS	5.1	3.7	5.2	2.5	2.7	2.5
walking	50.7	51.2	53.0	52.4	52.5	51.2
woman	74.4	84.3	88.7	142.0	133.6	145.2

Appendix B

Tensorflow Graphs

A Tensorflow subgraph to extract an interpolated ROI can be seen in Figure B.3. It includes scaling and rotation operations using roi_X (lower-left) as ROI input coordinates, and Tensorflow's built-in interpolate_bilinear at the top of the graph to generate interpolated pixel values from the incomming image (lower-right).

Figure B.4 shows incomming interpolation points for 300 regions-of-interest for corresponding particles (lower-left 300x87x107x2) across incomming image frame (lower-right 240x360). Here the heavy-lifting is performed by the interpolate_bilinear function available in Tensorflow.

Application of the likelihood function in Tensorflow is shown in Figure B.5. The incomming template can be seen in the lower left as thin lines. The incomming particle ROIs in the lower-right and represented as thick lines on the graph. The nodes on the right in the figure are storage locations for reusable pieces of the cross-correlation calculations performed by the einsum nodes. These calculations are given names starting with 'e' here.

Figure B.6 shows use of Tensorflow's multinomial resampling capabilities for generating resample indices for use in selection by a gather operation.

Template history and updates are maintained within the template_update subgraph as seen in Figure B.7. The highest particle likelihood

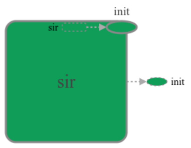


Figure B.1: Top level Tensorflow graph

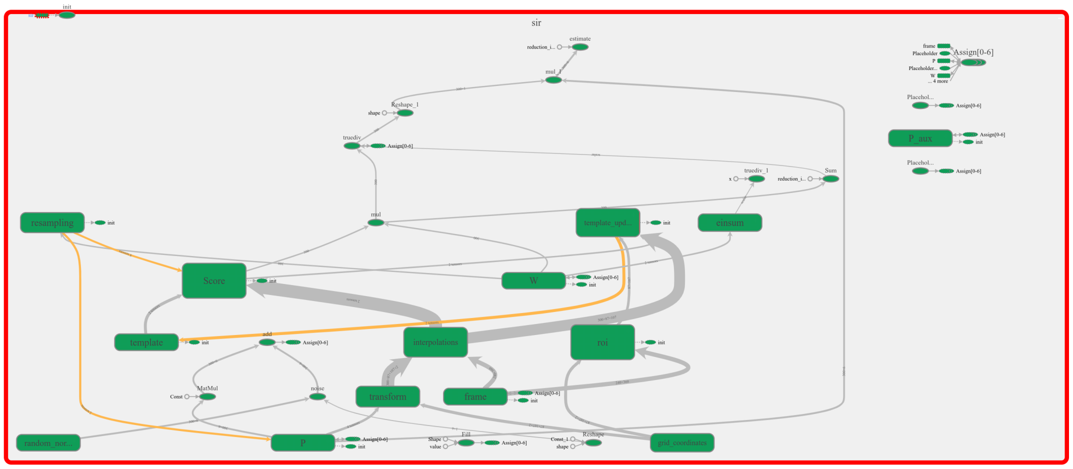


Figure B.2: Expanded Tensorflow graph

is determined by the ArgMax node here and is used to select from the incoming particle ROI interpolations seen entering from the lower-right.

Template updating is performed inside of the template_history subgraph of Figure B.8. Here there are graphs for the different modes of template updating. The ArgMax node selects the largest likelihood valued ROI from the template_history queue. SVD is performed by the Svd node, and as can be seen here is in red due to its incompatibility with Tensor Processing Units (TPUs). This causes the Svd to be performed on the CPU requiring the passing of the contents of template_history to main memory. The impact on speed is negligible as compared to the computational requirements of particle ROI interpolation and the likelihood function calculations.

Transformation of the template grid coordinates by the particle coordi-

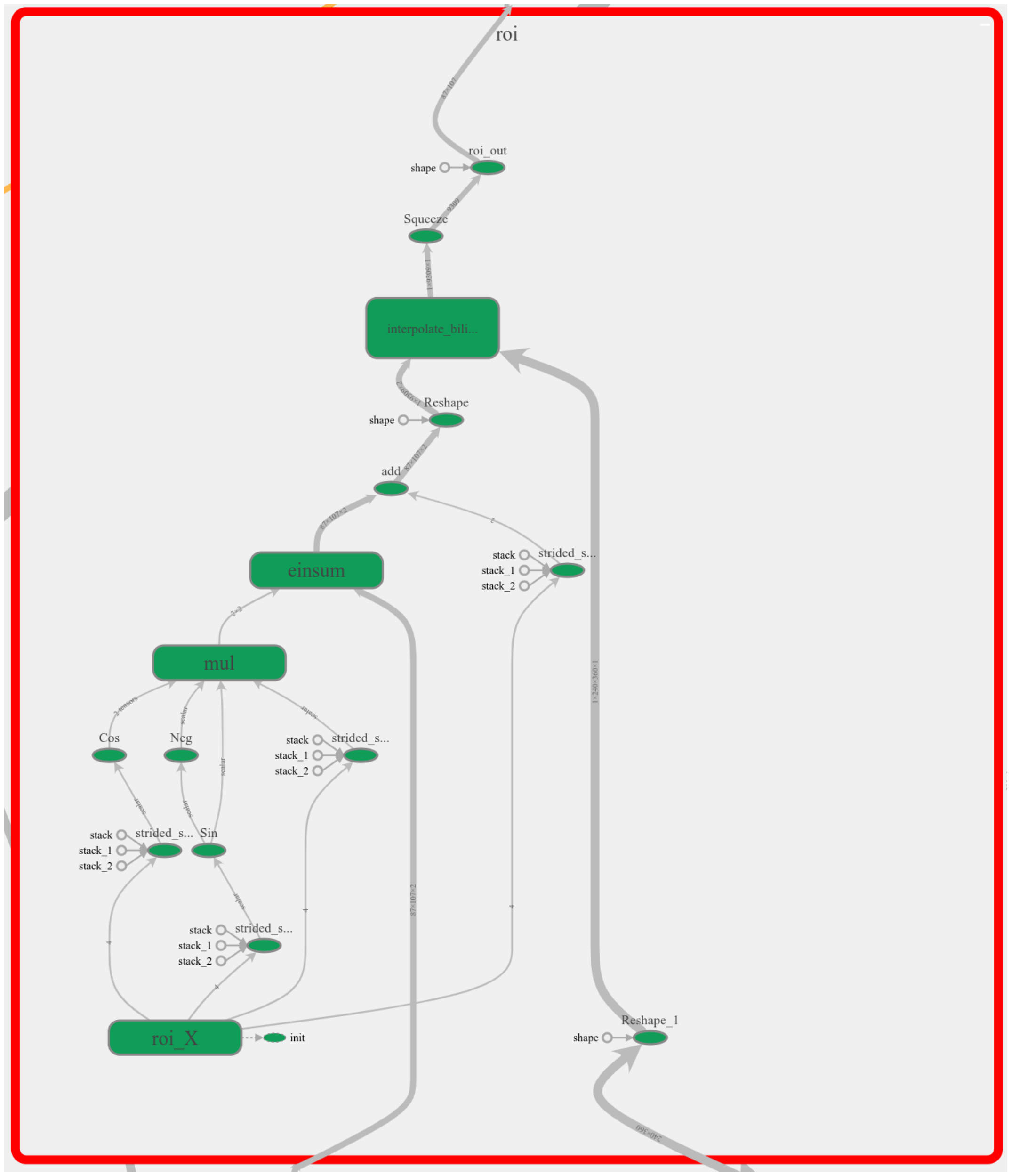


Figure B.3: Region-of-interest subgraph

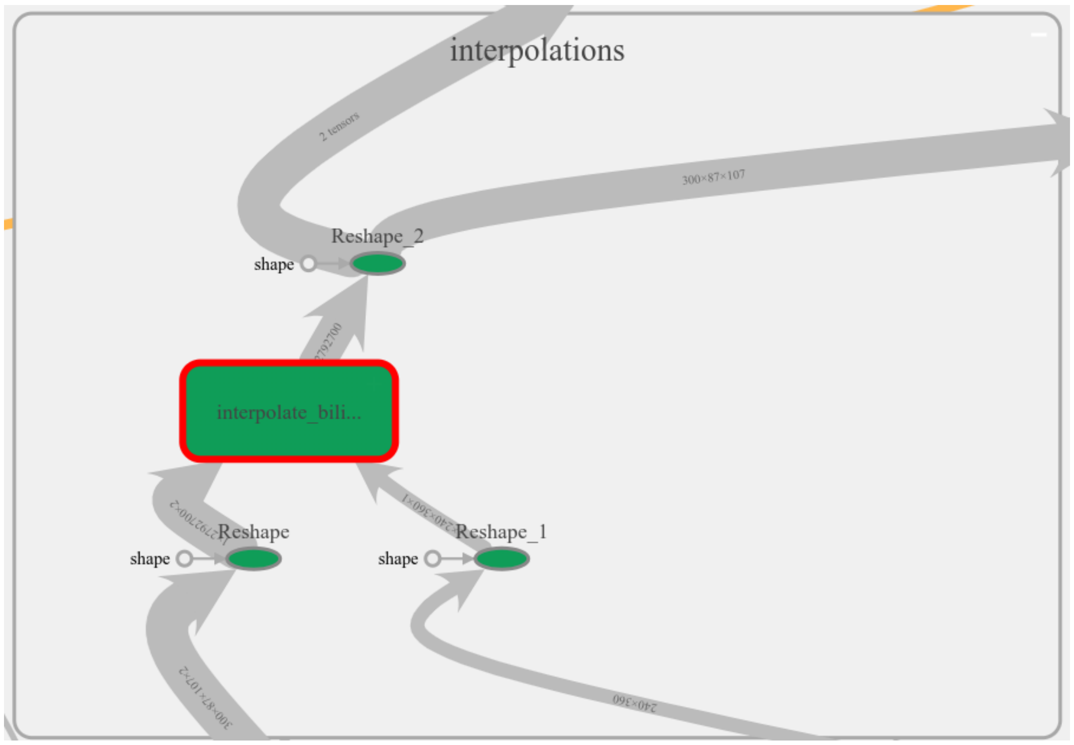


Figure B.4: Interpolations subgraph

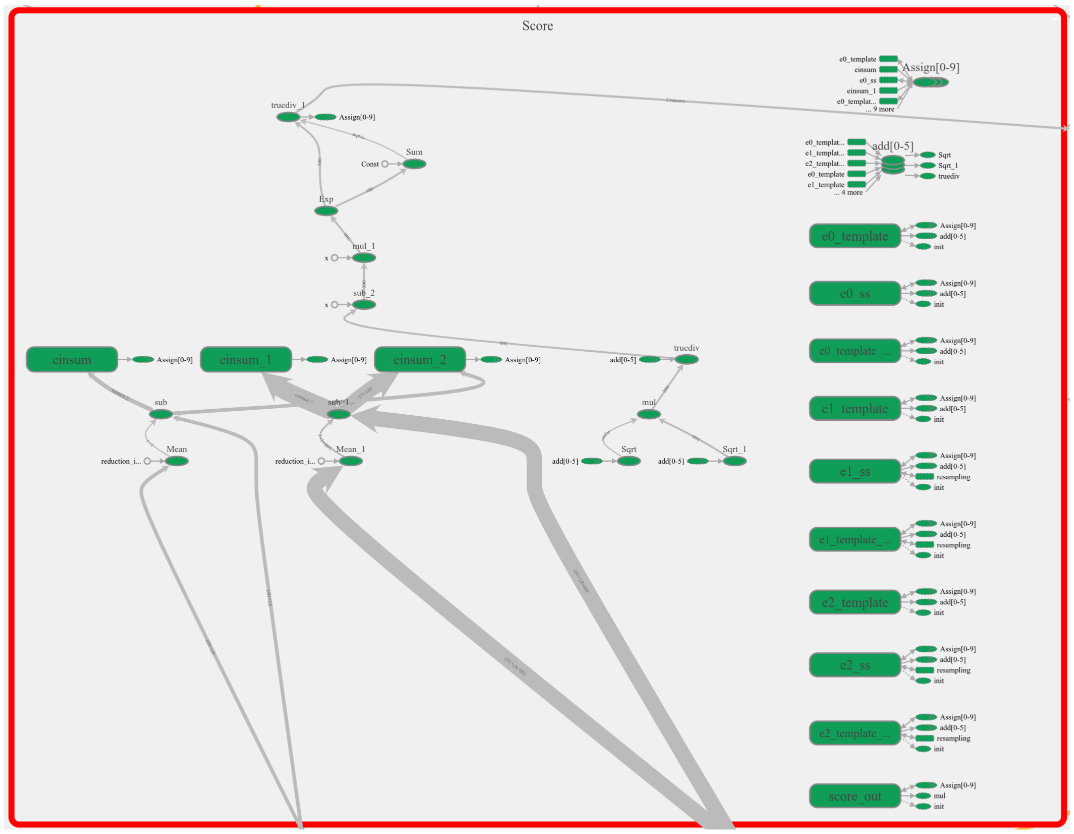


Figure B.5: Particle Likelihood function (scoring) subgraph

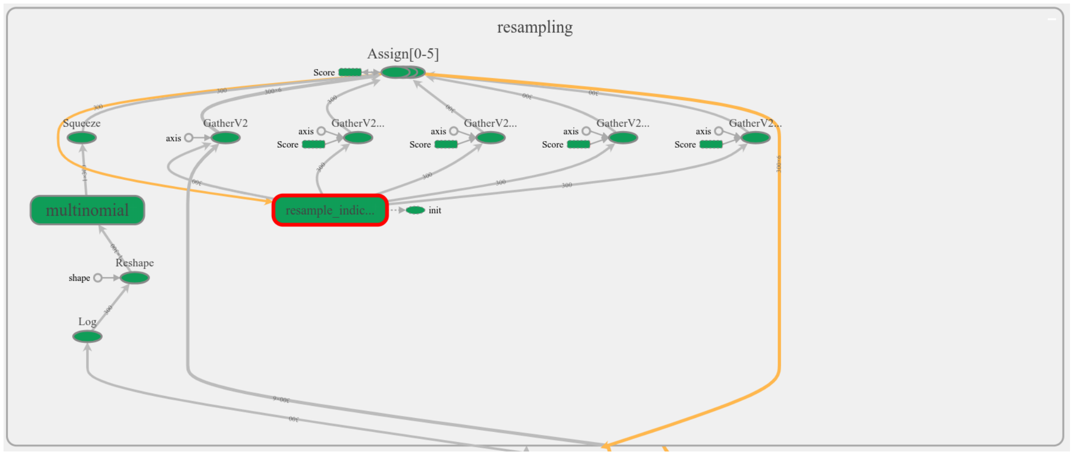


Figure B.6: Resampling subgraph

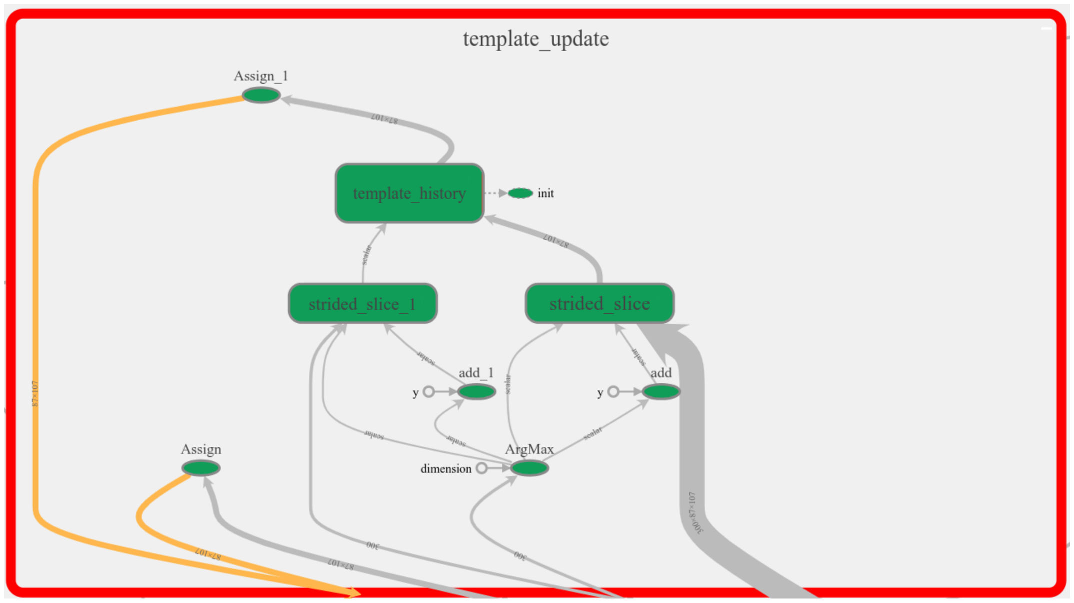


Figure B.7: Template History and Updating subgraph

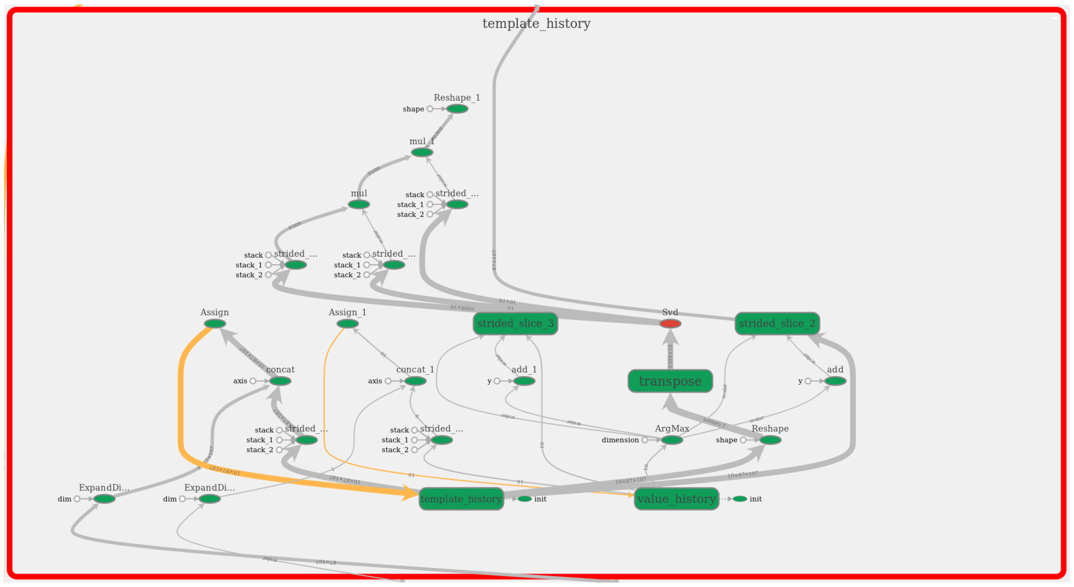


Figure B.8: Template History and SVD subgraph

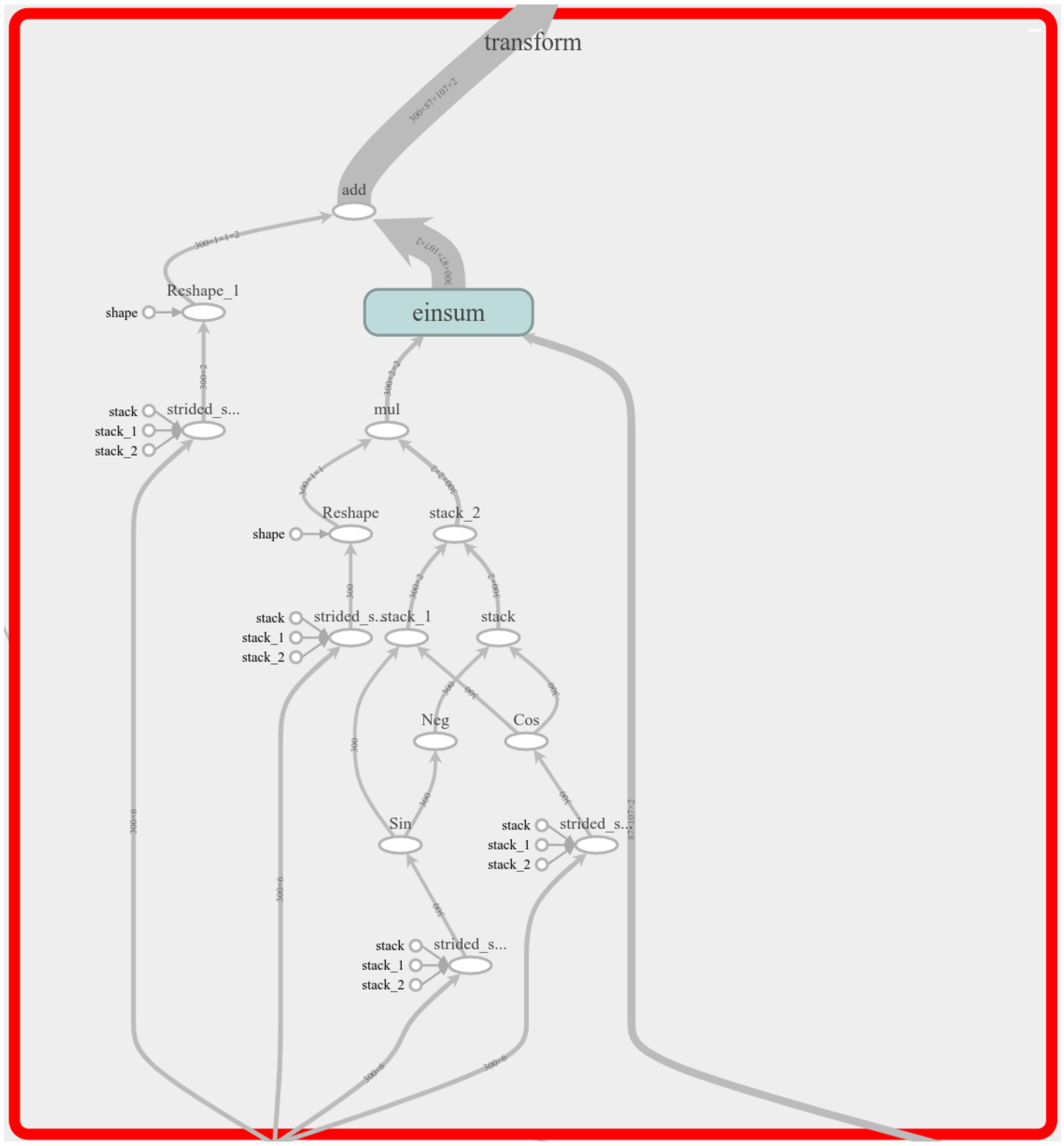


Figure B.9: Particle ROI points transformation subgraph

nates is shown in Figure B.9. The result of this ensemble calculation is then passed to the interpolations subgraph of Figure B.4.

Bibliography

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

[2] T. Bai and Y. Li, "Robust visual tracking with structured sparse representation appearance model," *Pattern Recognition*, vol. 45, no. 6, pp. 2390 – 2404, 2012, brain Decoding. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320311005048

[3] S. Baker and I. Matthews, "Lucas-Kanade 20 years on: A unifying framework," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, Feb 2004. [Online]. Available: https://doi.org/10.1023/B:VISI.0000011205.11775.fd

[4] J. C. F. Bello and J. P. Havlicek, "A state vector augmentation technique for incorporating indirect velocity information into the likelihood function of the SIR video target tracking filter," in *2016 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, March 2016, pp. 109–112.

[5] M. G. S. Bruno, "Sequential importance sampling filtering for target tracking in image sequences," *IEEE Signal Processing Letters*, vol. 10, no. 8, pp. 246–249, Aug 2003.

[6] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, May 2002.

[7] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–577, May 2003.

[8] A. Dawoud, M. Alam, A. Bal, and C. Loo, "Decision fusion algorithm for target tracking in infrared imagery," *Optical Eng.*, vol. 44, pp. 026 401–1–8, Feb. 2005.

[9] C. del Blanco, F. Jaureguizar, N. Garcìa, and L. Salgado, "Robust automatic target tracking based on a Bayesian ego-motion compensation framework for airborne FLIR imagery," in *Polarimetric and Infrared Infrared Processing for ATR*, ser. Proc. SPIE, F. Sadjadi and A. Mahalanobis, Eds., vol. 7335, 2009, 12 pp.

[10] A. Doucet, N. de Freitas, and N. Gordon, *An Introduction to Sequential Monte Carlo Methods.* New York, NY: Springer New York, 2001.

[11] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," *IEE Proceedings F - Radar and Signal Processing*, vol. 140, no. 2, pp. 107–113, April 1993.

[12] W. Guogang, Z. Zhijia, and W. Ying, "Research on SVD-based template-updating strategy," in *2006 5th IEEE International Conference on Cognitive Informatics*, vol. 2, July 2006, pp. 944–947.

[13] W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 34, no. 3, pp. 334–352, Aug 2004.

[14] Z. Huang, "An investigation of deep tracking methods," in *2017 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, Dec 2017, pp. 58–61.

[15] R. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME - Journal of basic Engineering*, vol. 82, pp. 35–45, 01 1960.

[16] J. Khan and M. Alam, "Efficient target detection in cluttered FLIR imagery," in *Optical Pattern Recog. XVI*, ser. Proc. SPIE, D. Casasent and T.-H. Chao, Eds., vol. 5816, 2005, pp. 39–53.

[17] T. Li, M. Bolic, and P. M. Djuric, "Resampling methods for particle filtering: Classification, implementation, and strategies," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 70–86, May 2015.

[18] X. R. Li and V. P. Jilkov, "Survey of maneuvering target tracking. part i. dynamic models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333–1364, Oct 2003.

[19] P. Malarvezhi and R. Kumar, "Particle filter with novel resampling algorithm: A diversity enhanced particle filter," *Wireless Personal Communications*, vol. 84, no. 4, pp. 3171–3177, Oct 2015. [Online]. Available: https://doi.org/10.1007/s11277-015-2793-4

[20] L. Matthews, T. Ishikawa, and S. Baker, "The template update problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 810–815, June 2004.

[21] L. Murray, "GPU acceleration of the particle filter: the Metropolis resampler," 2012.

[22] C. T. Nguyen, J. P. Havlicek, G. Fan, J. T. Caulfield, and M. S. Pattichis, "Robust dual-band MWIR/LWIR infrared target tracking," in *Proc. 48th Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, Nov. 2-5, 2014, pp. 78–83.

[23] C. F. Olson, "Maximum-likelihood image matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 6, pp. 853–857, June 2002.

[24] K. Pugalenthi and N. Raghavan, "A holistic comparison of the different resampling algorithms for particle filter based prognosis using lithium ion batteries as a case study," *Microelectronics Reliability*, vol. 91, pp. 160 – 169, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002627141830814X

[25] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You Only Look Once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[26] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, June 2017.

[27] R. Rohilla, V. Sikri, and R. Kapoor, "Spider monkey optimisation assisted particle filter for robust object tracking," *IET Computer Vision*, vol. 11, no. 3, pp. 207–219, 2017.

[28] M. Shook, J. Junger, N. Mould, and J. P. Havlicek, "Quantifying infrared target signature evolution using AM-FM features," in *2010 IEEE Southwest Symposium on Image Analysis Interpretation (SSIAI)*, May 2010, pp. 189–192.

[29] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1442–1468, July 2014.

[30] A. F. M. Smith and A. E. Gelfand, "Bayesian statistics without tears: A sampling-resampling perspective," *The American Statistician*, vol. 46, no. 2, pp. 84–88, 1992. [Online]. Available: http://www.jstor.org/stable/2684170

[31] L. A. Turner and C. H. Sherlock, "An introduction to particle filtering," 2013.

[32] V. Venkataraman, G. Fan, J. Havlicek, X. Fan, Y. Zhai, and M. Yeary, "Adaptive Kalman filtering for histogram-based appearance learning in infrared imagery," *IEEE Trans. Image Process.*, vol. 21, no. 11, pp. 4622–4635, Nov. 2012.

[33] L. Wang, T. Liu, G. Wang, K. L. Chan, and Q. Yang, "Video tracking using learned hierarchical features," *IEEE Transactions on Image Processing*, vol. 24, no. 4, pp. 1424–1435, April 2015.

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

[35] S. Yi and L. Zhang, "A novel multiple tracking system for UAV platforms," in *ISR Systems and Applications III*, ser. Proc. SPIE, D. Henry, Ed., vol. 6209, 2006, 8 pp.

[36] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, Dec. 2006. [Online]. Available: http://doi.acm.org/10.1145/1177352.1177355

[37] J. Zhang, L. Yang, and X. Wu, "A survey on visual tracking via convolutional neural networks," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, Oct 2016, pp. 474–479.