UNIVERSITY OF CALIFORNIA,
IRVINE

A Visual Tracking Study and A Proposal of Modifications

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Mechanical Engineering

by

Yu Hua Nicole Tseng

Thesis Committee:
Professor Athanasios Sideris, Chair
Professor Tryphon T. Georgiou
Professor Haithem Taha

2019

DEDICATION

To my mother, father, and my two sisters,
who supported me unconditionally on all fronts along the way.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my sincere gratitude for Professor Athanasios Sideris, whose guidance, patience and encouragement made the work possible.

I would also like to express my deepest appreciation for the committe members, Professor Tryphon T. Georgiou and Professor Haithem Taha, for their kind support.

Finally, I would like to express my gratitude for my family and Ronald Daries for the boxes and boxes of Studentenfutter of all varieties.

ABSTRACT OF THE THESIS

A Visual Tracking Study and A Proposal of Modifications

By

Yu Hua Nicole Tseng

Master of Science in Mechanical Engineering

University of California, Irvine, 2019

Professor Athanasios Sideris, Chair

On-line visual tracking of a specified target in motion throughout frames of video clips faces challenges in robust identification of the target in the current frame based on the past frames. Three approaches for tracking the target image patch are described and compared. These approaches utilize particle filtering and principal component analysis (PCA) to identify the most likely location of the target in the current frame and a low dimensional subspace representation of the patches of images to be kept as the templates in the dictionary for the identification. By using a combination of methods and compare the result of each, a new model based is proposed. The goal is to achieve a more robust and accurate tracking of a target throughout the video and continue updating the identification templates to adapt the target changes, such as apparences in lighting, angle, scale and occlusions. The challenges in tracking are to introduction of the "right" templates into the identification templates in the dictionary and identify the most accurate particle image patch while tracking the target with the right tracking patch scaling.

The first approach considered and on which the structure of the visual tracker is based is the "Incremental Learning for Robust Visual Tracking" by D. Ross et al., which is a computationally fast tracker that utilizes a method of low dimensional subspace for the identification template dictionary and incremental PCA for its tracking. The tracker has a

simple rule in accepting the patches of images to be in the identification template dictionary after the image patch has gone through a singular value decomposition (SVD), where it eliminates singular values are smaller than 10^{-6} of the sum of squared sinuglar values and the corresponding bases are also eliminated. This elimination scheme has very limited robustness in tracking, therefore, more selective processes in accepting identification templates in the dictionary are explored and introduced on top of the existing method in comparison and to address the challenges in on-line video tracking.

The second approach is the "Least Soft-Threshold Squares Tracking" proposed by D. Wang et al. solves the least soft-threshold squares distance problem to identify the distances of the particles to the templates in the dictionary, which greatly improves the tracking accuracy. This method is also computationally cheap in comparison to the first approach, and its accuracy is also better than the first approach, but it would sometimes fail to track in some applications. Finally, the third approach reviewed is the "Robust Visual Tracking and Vehicle Classification via Sparse Representation" by X. Mei et al. is to weight each particles when selecting the most likely target patch so the best patch has a highest weighted probability which ensures it being selected and introduced to the template dictionary. This approach performs well in comparison to the first and the second approaches in tracking accuracy and robustness, but this approach is extremely computationally expensive.

Three new components are proposed in an effort to mitigate some of the limitations that the three approaches exhibit. One such component is to simply reject the image patches that exhibit too great of difference to the current template dictionary, which resulted in improved tracking robustness. This method is computationally cheap and easy to implement. Another component introduced is a second set of dictionary that is composed of admitted image patches, which is used for tracking when the image patches appears to be too dissimilar to the dictionary with low dimensional representation. It is expected that with more well defined and stronger features, it forces the tracking to identify the target. Finally, the third

component introduced is the to prevent shrinkage of the target boundary box by weighting the particles drawn with the ratio of area change so that more weight is placed on particles with less arial change. This increases the likelihood of recovering the target again if tracking loses the target, and instead of shrinking the boundary box, the tracking is biased to staying with the image patch of the same size. The resulting performance of the proposed tracking scheme has not been noticeably improved, part of the reason is because the metrics available to identify a noisy image patch from the good image patches are not always indicative of the noisy-good image patch divide.

Chapter 1

Introduction

Visual tracking requires robustness and speed. Robustness comes from the tracker's ability to accurately identify the specified target consecutively, and the tracker's relates to the amount of computational process the tracking requires from frame to frame. Here, more emphasis is placed on the analysis on the robustness of the tracker rather than the speed, as computational speed also depends on the computing hardware and there is continued the advancement in its development.

The two main objectives of a visual tracker are to accurately identify the specified target and "predict" its motion/perturbation [33]. To track a target in motion, a few considerations and decisions have to be made for the tracker to deal with target appearance change and the target motion from one moment to the next. The target appearance may be affected by noise, scaling, occlusion, illumination variations, viewing angles and surrounding background [18], [13]. In fact, the main challenge in visual tracking lies in handling the variability of target appearance [18]. The target object motion/perturbation poses another challenge in visual tracking. A tracker's ability to estimate the current position of the target based on prior target positions is also a main component in judging the performance of a tracker [33].

A visual tracker must be flexible enough to track successfully when encountering statistically probable cases. Both of the main objectives of a visual tracker here are modeled statistically using Gaussian distributions. In the case of target appearance changes, the assumption is that the appearance of the target would be similar to that of the previous observations. A similar assumption is made with the motion of a specified target, where the target is assumed to move continuously and gradually within the confine of the frame from one instance to the next, rather than disappearing from one location and reappearing far away within the frame from one moment to the next.

1.1 Visual Recognition

There are two general representation models that visual trackers take on: global and local [24]. Global exploits overall observation that varies in the the whole target region, analyzing the target image patch as a whole image when comparing target image patches. Global representation reflects the information of all references, e.g. low dimension representations. For instance, when there is change in illumination, it is reflected on the global appearance change. Whereas local treats the target image as many small localized image patches in order to compare the changes in each small region. For instance, when a partial occlusion occurs in the image, it is captured locally, but not globally, so that other localized regions are still able to capture parts of the target appearance.

There are broadly two types of pattern recognition, discriminative and generative [30], [5], [22], [9]. A discriminative model is a conditional probabilistic modeling combining structure, priors, invariants, latent variables and data to form a good joint density of the domain as a whole that directly optimizes a relatively less domain-specific model only for the classification or regression mapping [5]. Assuming the conditional probability of the information Y given the observation Y, $P(X|Y)$, and estimate the data directly using the conditional probability

$P(Y|X)$. A generative model is a statistical model that assumes within this generative density, one can specify partial knowledge a priori and refine this model with incomplete information using empirical observations and data. Given the probability of the target $P(Y)$ and the conditional probability distribution of the observation X given the target Y, $P(X|Y)$, through training and using Bayes rule, the conditional probability of the target Y given the observation X, $P(Y|X)$, can be found.

To track the target in a frame, the visual tracker would need some reference to identify the target. The reference has to contain enough information about the appearances of the target for a visual tracker to successfully perform. The reference functions like an image dictionary for the target, and these target images need to contain the right amount of target appearance information [16]. The large amount of images about target appearances can be stored efficiently and compactly in a low-dimension subspaces, such as in eigenspace or sparse subspace, where multiple images could be encoded into and represented by the subspace. Two examples of these low-dimension subspace dictionaries for storing images in visual tracking applications, eigenspace and subspace dictionaries, are further discussed here.

1.1.1 Discriminative Model

Discriminative models require some references to differentiate the intended target from the rest of the image because it is structured to classify items. Some discriminative computer vision require off-line training and testing prior operation and some do not.

If training is required, it is recommended to train on large amount of training data sets to allow refinement of classifications. The quality of the training data sets influences the ability of the discriminative model to identifying objects [14]. When encountering an object with an appearance that has not been trained on, the recognition runs the risk of identification

failure. The advantage of having already been trained on data sets is that the recognition performance would demonstrate high accuracy. Trained discriminative models have gained wide popularity since the introduction of convolutional neural network (CNN) from the breakthrough of LeNet-5 [11], which achieved a high accuracy visual recognition using CNN to AlexNet [8] that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) of 2012. Notably, GoogLeNet [26] won the ILSVRC of 2014, and, most recently, NASNet [35] developed by Google Inc. exceeded previously achieved accuracy in computer visual recognition.

Visual trackers could be built to take advantage of the benefits that CNN offers. This is demonstrated by Apostolopoulos et al. [1] where their visual tracker can track a target that is partially occluded by recognizing the object as the specified target. Many others have demonstrated ability to visually identify targets with high accuracy and speed [1], where the tracker proposed by Held et al. [4] runs at reportedly 100 fps. In recent years, discriminative computer vision has demonstrated a near-human visual recognition and currently dominates computer vision/recognition development.

A discriminative visual recognition without training still requires a references for differentiating targets from non-targets. An approach is learning on the go by collecting the target image patch in a reference template dictionary. This approach provides flexibility in the choice of targets and in responding to target appearance change, as it is independent of the training data sets. The drawback without pre-training is that if the target appearance is compromised due to noise, scaling, occlusion, illumination variations, viewing angles and surrounding background, the learning in the reference dictionary is corrupted as well.

The visual tracker by Shen et al. [22] incrementally learns both the appearance of the target and the background and separates them by collecting a discriminative sparse subspace dictionary. This tracker has demonstrated competitive robust tracking in the criteria of center location error (CLE), overlap rate (OR) and success rate (SR) running reportedly above 33

frames per second (fps). Another work by Sun et al. [25] implemented incremental sparse subspace discriminative dictionary learning of the target appearances also demonstrated good tracking performance in the criteria of CLE and OR.

1.1.2 Generative Model

A generative model-based visual recognition also requires references of the target appearance to estimate similarity of the given image to that in the reference. As the case of discriminative models, the reference could be acquired by training or adopted incrementally, and each method has its associated advantages and disadvantages as discussed in section 1.1.1.

A trained generative CNN visual tracker by Song et al. [23] uses a generative adversarial network (GAN). They argue that discriminative features tend to over fit in one frame and while the features available may not be robust, and generative adversarial learning provides more robust features and mitigates the over-fitting in one frame. This tracker demonstrates competitive performance with the ability to accurately track the target at, reportedly, 1.5 fps. The generative visual tracker proposed by Zhou et al. [34] demonstrates an incremental learning tracker that is able to track a target more accurately in the Pattern, Analysis, Statistical Modelling and Computational Learning (PASCAL) overlap rate criterion than the benchmark trackers at the time, which were all discriminative models. Despite the robustness of this tracker, it reportedly runs only about 5 fps.

1.1.3 Low Dimensional Subspace Representation

Visual tracking must tackle the issue of target appearance variability throughout the period of tracking. As briefly mentioned in section 1.1.1, there are two approaches: pre-training the visual tracker and adapting to appearance changes incrementally.

Using eigenspace to represent and learning an image is proposed by Hiroshi et al. [16], in which eigenspace representations are in a compact low-dimension approximated encoding of a large set of images. The compact low-dimension approximations are in terms of a small number of orthogonal eigenbasis images. The eigenbasis images span a subspace of the training images, i.e. eigenspace, and a linear combination of them is used to approximately reconstruct any of the training images. This is an efficient way of storing appearances of one object, such as the appearances of the object under different illuminations [2] and viewing angles.

Two observations are made by Black et al. [3] on eigenspace representation technique. First, since this technique requires a least-squares fit between an image and the eigenspace, the eigenspace would have a poor representation when there are noises in the input images being matched. Secondly, eigenbasis representation tries to represent all possible angles of an object and all of its possible appearances, but it is easier to represent a smaller set of canonical appearances allowing a parameterized transformation, e.g. affine, to link the input images and the eigenspace. This involves appearance estimation and transformation of the parameters that takes the appearance into the image, resulting a multiple views and a transformation model.

Black et al. [3] proposed tracking by pre-training eigenbasis representations, a low-dimension subspace, and a robust error norm, which assumes an eigenspace subspace constancy for motion estimation. The robustness of their visual tracking requires a large amount of training data that must include all possible appearance variations of the tracked target, i.e. all appearance variations of illuminations and angle perspectives, to form the representation eigenbasis. Though this visual tracker is dependent on the training data set and is inflexible to the tracking target once it is trained on one particular object, it demonstrates the potential for an appearance reference to be stored in a low-dimension subspace.

Another low-dimension subspace technique has since been developed in visual recognition to

efficiently store templates in the dictionary using the "least absolute shrinkage and selection operator" (LASSO) method [27]. The LASSO method is a sparse representation technique by solving an optimization problem of an input image to a least-squares optimizing eigenspace, i.e. 2-norm ℓ_2 space, with a 1-norm ℓ_1 space problem. Depending on the type of problem being posed, the intended solution for the optimization is different, but the goal is to promote the sparsity in the least-squares problem. Wright et al. [32], [31] proposed using sparse representation for human face and object classification, which produced lower classification errors. Kumar et al. [9] proposed a generative visual tracker with robust sparse representation and weighted principle component analysis (PCA), which has performed competitively with many bench mark visual trackers in the criteria of center location error (CLE) and overlap rate (OR) at a running rate above 1.6 fps.

1.2 Motion Tracking

The challenge in target tracking in a video frames is that there are cases when the motion of the target is not smooth in relation to the confinement of the frame. This is due to the interactions between the motions of the camera and that of the target. For instance, if the objective of the camera that is not stationary is to keep the target near the center of the camera scope frame, then the camera frame would try to adjust to the target motion as the target moves. Though the movement of the target is smooth relative to Earth, in relation to the camera scope frame, the motion of the target could be erratic due to the camera adjustment. Another occurrence could be that the camera frame itself is moving unsteadily due to the camera being mounted on another moving object, in which case, the target motion moving in the frame would also result to be unpredictable. This phenomenon is illustrated by tracing the motion of one spot on the target in two video clips in Figure 1.1a and Figure 1.1b, where a spot on the target object moves about within the confine of the frame in a non-

smooth manner and often moves erratically with many sharp direction changes.

These non-smooth and unpredictable motions of the target relative to the video frame illustrate the difficulties to implement a model for all cases of target motion that could predict with good accuracy based prior frames, e.g. a smooth motion model encountering a case of predicting a low probability of a target moving right in the current frame, which is based on knowing that it moved left in the prior frames, even though the target does move to the right. Despite the potential unpredictability of the motion, it is assumed that the target motion is continuous and incremental, where the current position of the target in the frame is going to be near the previous position. Therefore, for a visual tracker to track the target position relative to the frame, the tracker must search all vicinity around the previous target position relative to the frame. An approach that allows this type of search is particle filtering.

(a)

(b)

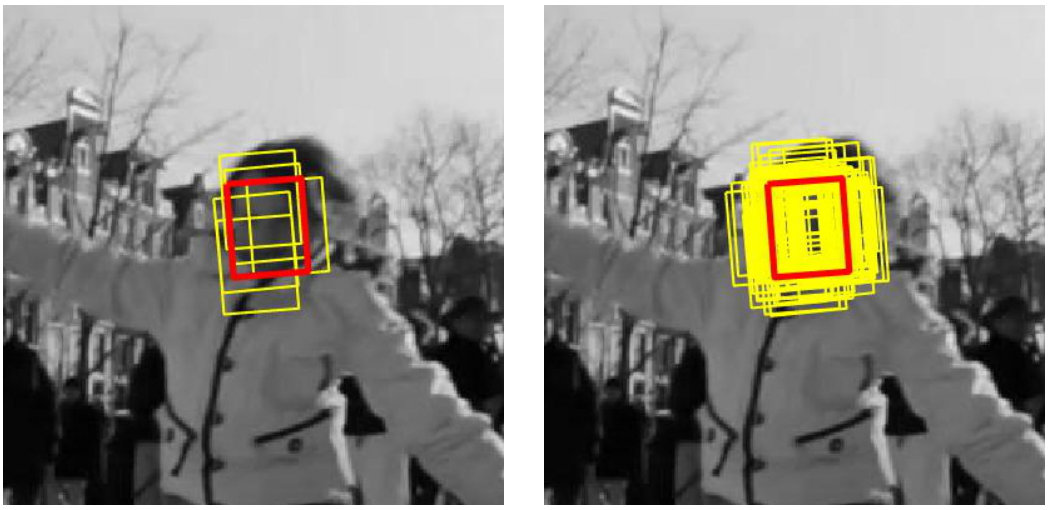
Figure 1.1: Examples of the movement (yellow line) of a spot (red cross, ×) on a target within the confine of the frame throughout the video clip.

1.2.1 Particle Filter

In the application of computer visual tracking, to estimate the position of the target in the current frame given the prior frame, trackers must search near the position of the target in the prior frame. With the assumption that the target behaving with Brownian motion, particle filtering achieves the target position estimation by drawing particles based on affine parameters, i.e. location, scaling and angle. This assumes a Gaussian probabilistic distribution of the parameters, each with an independent variance. Each parameter of the target

particle is drawn with a random value based on its respective variance, and the parameters are grouped in a set where each parameter describes a different aspect of the particle to form a state of the particle. The particle states can be used to transform into corresponding images patch particles, where each image patch particle has slightly perturbed parameters, e.g. slight position translation, adjusted scaling and/or slight rotation from the prior frame.

The more particles are drawn, the more likely one of the particle image patches captures the target in the current frame allowing the tracker to successfully track the target. This is illustrated in examples shown in Figure 1.2 that compares a case of drawing 5 particles and a case of drawing 75 particles, where the case with 5 particles has lower likelihood in capturing the target in the current frame and the case of drawing 75 particles has much higher likelihood in capturing the target in the current frame. Although it should be noted that more particles will be more computationally expensive in identifying the target for each frame. This is a trade-off between the efficiency, i.e. number of particles drawn, and effectiveness, i.e. ability for particle filter to estimate the posterior distribution.



(a) 5 image patch particles drawn. (b) 75 image patch particles drawn.

Figure 1.2: Examples of image patch particles drawn (yellow rectangles) with respect to the identified target image patch of the previous frame (red rectangle).

As discussed in 1.2, due to unpredictability of the target motion direction, it is difficult to

implement a general motion model to predict the actual direction of target motion based on target motion of prior frames, therefore, particles could not be drawn from a predicted location. But by assuming that target behaves in Brownian motion, the unpredictability could be compensated by drawing more particles in a larger parameter search region.

A possible limitation of implementing a particle filter in visual tracking may be that the speed of target being faster than predicted, where the variances are not large enough, resulting the search region not large enough to capture the target, i.e. the variances for the x and y translation parameters being too small. Another possible limitation is that the the search region of the parameters is large enough, but there are not enough particles to identify the target with good accuracy, resulting the identification of a target image patch not being representative of the target appearance.

Regardless of the type of representation model or the type of probabilistic model of the visual trackers, particle filtering has been a popular choice to estimate target image patch demonstrating good performances, e.g. Ross et al. [18] and Song et al. [23].

1.3 Related Work and Motivation

This work is motived by the Incremental Visual Tracking (IVT) proposed by Ross et al. [18], which is a global discriminative approach to identify the target image patch consecutively with an incrementally adaptive template dictionary. A particle filter is implemented in IVT to address the motion tracking. IVT uses gray-scale images to conduct the tracking. The dictionary is a set eigenbasis vectors forming low dimensional image representation subspaces, and a series of dictionary templates are used for identifying the target and, at the same time, being adaptive to the changing appearance of the target during the tracking. Each template in the dictionary represents a low dimensional PCA subspace, i.e. a PCA eigenbasis vector,

and the errors are calculated by an ordinary least squares method. With the implementation of these ideas, IVT has the the advantage of being trained on-line without any off-line pre-training or structural tuning, thus, allowing it to be independent of the quality and availability of data sets.

Two other low-dimension subspace approaches are explored here to address the challenges in identifying the target image patch and updating the template dictionary. These two tracking approaches are chosen due to their similarity in the tracking scheme where they also keep an incrementally adaptive dictionary of templates that represent prior target images, their use of particle filtering for motion tracking and that they also use gray-scale images in tracking. One approach is a global generative approach called "Least Soft-Threshold Squares Tracking" (LSST). LSST is proposed by Wang et al. [30], which, in addition to the eigenbasis templates forming its dictionary, solves for the sparse noise from a 1-norm ℓ_1 and 2-norm ℓ_2 optimization problem. The resulting solution is the least soft-threshold square distances to the template dictionary as a metric for identifying the most probable target image patch particle in the current frame. This approach tries to identify the outliers effectively by separating noise into sparse and non-sparse parts. The other approach is "Robust Visual Tracking and Vehicle Classification via Sparse Representation" (RVT) proposed by Mei et al. [13], which solves for a 1-norm ℓ_1 regularized least-squares problem to find the target coefficients associated with every image patch particle and use the coefficients to weight the image patches introduced into the dictionary. The tracker runs reportedly only at 0.5 fps due to having to solve the computationally expensive optimization problem.

Chapter 2

Incrementally Adaptive Reference Approaches

As discussed in section 1.1.1, off-line trainings for a visual tracker require a large amount of data sets and the tracking results are dependent on the quality of the training data sets, i.e. object varieties and all their possible appearances. A solution is to keep an appearance template dictionary. But an issue arises with the appearance templates when the target appearances change drastically over the tracking period and the templates do not reflect the changes, e.g. when tracking a human head as a target where the features of a face are the initial target appearance, but later in the tracking, the head is turned to have the appearance of the back of the head. To address this temporal appearance issue, Incremental Visual Tracking (IVT) proposed by Ross et al. [18] allows the appearance templates in the dictionary to be adaptive. The dictionary is updated with the target image patches throughout the tracking period, so that the dictionary always reflects both the past and the current appearances of the target, making the tracking more robust. The incrementally adaptive dictionary has demonstrated good results to robustly track targets as demonstrated by Shen et al. [22], Sun et al. [25] and Zhou et al. [34], among many others.

In the following sections, three incremental adaptive reference tracking approaches are discussed in detail: Incremental Visual Tracking (IVT), Least Soft-Threshold Squares Tracking (LSST) and Robust Visual Tracking (RVT). For the purpose of comparing these three approaches, they are assumed to draw particles the same way with inference Markov hidden parameters and are able to transform a set of image patch parameters into the image patches, and that the dictionary each keeps is in the subspace spanned by eigenbasis and centered at the basis mean or by the previously-seen image patches. These mechanisms that are assumed to be the same in the three tracking approaches are elaborated in the discussion of IVT and are referenced to by the subsequent discussions on LSST and RVT.

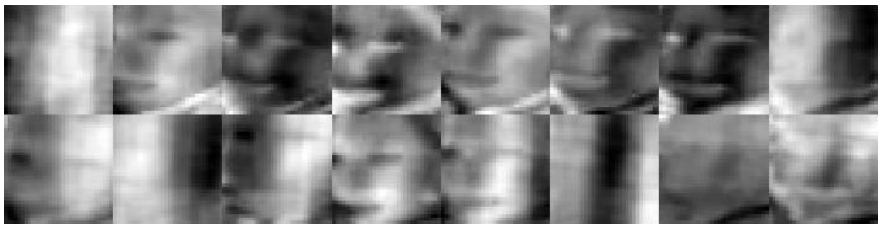
With an incrementally adaptive dictionary, the tracker has to decide which target image patches to populate the dictionary with: which image patches should be accepted and retained in the dictionary and what traits would constitute a good target image? Each visual tracker has an approach in selecting the target images in its particle filter and metrics in selecting the chosen target image patches into the template dictionary. These decisions and metrics are discussed at length in the following sections.

After the decision of whether or not to accept the chosen target image patch into the dictionary, the tracker structure has to wrestle with another decision: the number of templates to be kept in the dictionary. For instance, in the case of image patch template dictionary, if there are too many templates in the dictionary, meaning that much earlier target appearances are remembered in detail, the dictionary may not reflect the more recent target appearances and may be less effective. But if there are not enough reference templates in the dictionary and the dictionary remembers too little information, and the dictionary may not be sufficient to identify the target after an occlusion. The focuses for the tracking performances are empirically analyzed in chapter 3 where experimental results are presented.

An example of comparatively good eigenbasis templates in a dictionary and a corrupted one can be seen in Figure 2.1. The dictionary in Figure 2.1a exhibits much more similarities

among the templates where each template exhibit more distinct identifiable features, whereas the dictionary in Figure 2.1b has many templates that are dissimilar to one another and have faded/blurred features. Having faded features in the dictionary templates could mean that the dictionary has obtained image patches that are less representative of the target appearance, and the target representations in the eigenbasis templates have been changed by accepting those less representative image patches. With corrupted dictionary templates, the dictionary would be far less effective as a reference for target tracking, especially after an occlusion.

(a) An example of a comparatively good batch of eigenbasis templates in the dictionary.



(b) An example of a comparatively bad batch of eigenbasis templates in the dictionary.

Figure 2.1: The eigenbasis templates in the dictionary.

2.1 Incremental Visual Tracking (IVT)

Incremental visual tracking (IVT) proposed by Ross et al. [18] visually tracks a target without prior training, i.e. is independent of external training image data sets. They achieved this by implementing an incrementally adaptive reference dictionary that obtains the estimated target image patches during the tracking, which includes more recent appearances

in the dictionary to handle varying target appearances. Furthermore, the templates in the dictionary are eigenbasis representations of target image patches seen so far, and besides being an efficient way of storing images, they help collectively provide a compact notion of the target object being tracked [3] [10], [18].

The images IVT tracks are in gray-scale, and the target image patches are converted into a specific image resolution that are usually smaller than the actual image bounded by the tracking boundary box, e.g. 32 pixels-by-32 pixels. These image patches are stored and analyzed in this resolution during the tracking. From the proposal of Black et al. [3], as discussed in section 1.1.3, the target boundary box is defined by six affine parameters, and the image patch extracted is the image within the boundary box. More specifically, these boundary box affine parameters describe pixel translation in x, pixel translation in y, scaling of height and width, angle of rotation, aspect ratio, and skewness. Each pixel in the image patches is represented with a numerical value [0, 1]. This is because the image is in gray-scale, and per MATLAB definition, the numerical representation of the intensity of gray of each pixel is between 0 and 1, where 0 is pure black and 1 is pure white (other programming language like Python may use [0, 255] instead). Gray-scale scheme is different from red-green-blue (RGB) images, where each pixel on each layer represent intensity of red, green or blue, then the same pixel of each layer is combined to create a resulting color of mixing red, green and blue, each in its own intensity. In comparison, gray-scale images retains less information.

An initial target image patch on the first frame must be chosen before the tracking begins, and this image patch is also initialized as the dictionary template mean and is assumed to be from the zero-th frame. The particle filter draws the affine parameter particles from the estimated image patch parameters associated with the prior frame, each parameter is drawn with a pre-defined parameter variance. This produces particles that represent a Gaussian distribution within the mean. Then the corresponding image patches are extracted from

these affine parameter particles on the current frame. These images patch particles are in the pre-define resolution in numerical values that represent the gray-scale intensities of the pixels and are now ready for probability analysis. The probability of each particle indicates the similarity of the particle to the dictionary. An example of particles drawn based on a target image patch with respect to each parameter variance is illustrated in Figure 2.2. The random perturbation of the image patch parameters is to address the issues with target motion in respect to the frame discussed in 1.2 and with the variability of target appearances.

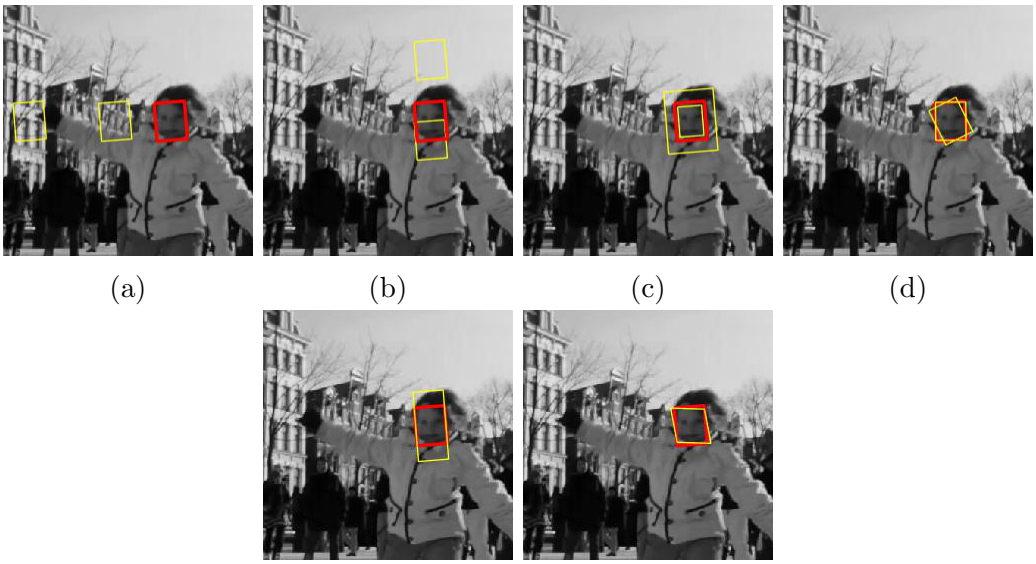


Figure 2.2: Examples of boundary boxes (yellow boxes) associated with particles drawn with only one single affine variance from the prior frame parameters (red boxes), each variance is for (a) translation of pixels in x, (b) translation of pixels in y, (c) scaling, (d) angle of rotation, (e) aspect ratio, and (f) skewness.

2.1.1 Incremental Principle Component Analysis (PCA): Updating the Dictionary And the Mean

The eigenbasis template dictionary is incrementally updated during tracking so that the dictionary reflect the appearance changes seen by the tracker thus far. In order for IVT to reference to the dictionary accurately, it is required that the eigenbases and their mean

being updated. The Sequential Karhunen-Loeve (SKL) procedures proposed by Levey et al. [12] is extended in IVT so that the algorithm updates not only the eigenbases, but also the basis mean. Ross et al. [18] call this algorithm the "incremental PCA algorithm."

The SKL Algorithm

Let a $d \times n$ data matrix be $\mathbf{A} = \{\mathbf{I}_1, ..., \mathbf{I}_n\}$, i.e. a populated dictionary, where $\mathbf{I}_i$ is a $d \times 1$ observation, i.e. an eigenbasis template in the pre-defined resolution and vectorized. The singular value decomposition (SVD) of $\mathbf{A}$ is computed so that $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$. When a $d \times m$ data matrix of new observations $\mathbf{B}$ is available, i.e. a batch of target image patches where each image patch is in the pre-defined resolution and is vectorized, the SVD of the concatenation of $\mathbf{A}$ and $\mathbf{B}$, $[\mathbf{A}\ \mathbf{B}]$, has to be computed efficiently into $[\mathbf{A}\ \mathbf{B}] = \mathbf{U}'\mathbf{\Sigma}'\mathbf{V}'^\top$. This provides updated eigenbases in $\mathbf{U}'$ and updated eigenvalues in $\mathbf{\Sigma}'$ so that have incorporated the new observation data in $\mathbf{B}$. The incremental PCA is only interested in finding $\mathbf{U}'$ and $\mathbf{\Sigma}'$, and they could both be computed by letting $\tilde{\mathbf{B}}$ be the component of $\mathbf{B}$ orthogonal to $\mathbf{U}$ so that $[\mathbf{A}\ \mathbf{B}]$ is partitioned as

$$[\mathbf{A}\ \mathbf{B}] = [\mathbf{U}\ \tilde{\mathbf{B}}] \underbrace{\begin{bmatrix} \mathbf{\Sigma} & \mathbf{U}^\top\mathbf{B} \\ \mathbf{0} & \tilde{\mathbf{B}}^\top\mathbf{B} \end{bmatrix}}_{\text{let be square matrix } \mathbf{R}} \begin{bmatrix} \mathbf{V} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}. \tag{2.1}$$

Let $\mathbf{R}$ be a square matrix of size $k + m$ and k is the number of singular values in $\mathbf{\Sigma}$, and that the SVD can be found so that $\mathbf{R} \overset{\text{SVD}}{=} \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^\top$ no matter what n is. With this, the SVD of $[\mathbf{A}\ \mathbf{B}]$ can then be expressed as

$$[\mathbf{A}\ \mathbf{B}] = [\mathbf{U}\ \tilde{\mathbf{B}}]\left(\tilde{\mathbf{U}}'\tilde{\mathbf{\Sigma}}'\tilde{\mathbf{V}}'^\top\right)\begin{bmatrix} \mathbf{V} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = \underbrace{\left([\mathbf{U}\ \tilde{\mathbf{B}}]\tilde{\mathbf{U}}\right)}_{\mathbf{U}'} \underbrace{\tilde{\mathbf{\Sigma}}}_{\mathbf{\Sigma}'} \underbrace{\left(\tilde{\mathbf{V}}^\top \begin{bmatrix} \mathbf{V} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}\right)}_{\mathbf{V}'^\top}, \tag{2.2}$$

where $\mathbf{U}' = [\mathbf{U}\ \tilde{\mathbf{B}}]\tilde{\mathbf{U}}$ and $\mathbf{\Sigma}' = \tilde{\mathbf{\Sigma}}$. Since the aim of SKL is to calculate $\mathbf{U}'$ and $\mathbf{\Sigma}'$, it is

not necessary to compute $\mathbf{V}'$ and it is left out of the computation in the SKL algorithm, i.e. $[\mathbf{A}\ \mathbf{B}] = [\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top\ \mathbf{B}]$ leaving out $\mathbf{V}'$ is instead expressed as $[\mathbf{U}\mathbf{\Sigma}\ \mathbf{B}]$. By following these procedures, an algorithm for SKL is as followed:

Given $\mathbf{A}$, from which the SVD is computed to get $\mathbf{U}$ and $\mathbf{\Sigma}$:

1. Taking the QR decomposition of $[\mathbf{U}\mathbf{\Sigma}\ \mathbf{B}]$ to obtain $\tilde{\mathbf{B}}$ and $\mathbf{R}$: $[\mathbf{U}\mathbf{\Sigma}\ \mathbf{B}] \overset{\text{QR}}{=} [\mathbf{U}\ \tilde{\mathbf{B}}]\mathbf{R}$.
2. Computer the SVD of $\mathbf{R}$ to obtain $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{\Sigma}}$: $\mathbf{R} \overset{\text{SVD}}{=} \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^\top$.
3. $\mathbf{U}' = [\mathbf{U}\ \tilde{\mathbf{B}}]\tilde{\mathbf{U}}$ and $\mathbf{\Sigma}' = \tilde{\mathbf{\Sigma}}$, and if the desired number of eigenbasis vectors in $\mathbf{U}'$ is less than the number of non-zero singular values, then these excessive vectors and singular values may be discarded.

Figure 2.3: The SKL algorithm.

The Incremental PCA Algorithm

The following lemma is in extension to allowing the computation of the updated basis mean:

Lemma 1. Let $\mathbf{A} = [\mathbf{I}_1, ..., \mathbf{I}_n]$ and $\mathbf{B} = [\mathbf{I}_{n+1}, ..., \mathbf{I}_{n+m}]$ be data matrices and $\mathbf{C} = [\mathbf{A}\ \mathbf{B}]$ be their concatenation. Denote the means and scatter matrices of $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ as $\bar{\mathbf{I}}_A$, $\bar{\mathbf{I}}_B$, $\bar{\mathbf{I}}_C$ and $\mathcal{S}_A$, $\mathcal{S}_B$, $\mathcal{S}_C$ respectively. It can be shown that $\mathcal{S}_C = \mathcal{S}_A + \mathcal{S}_B + \frac{nm}{n+m}\left(\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_A\right)\left(\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_A\right)^\top$.■

In this lemma, a scatter matrix is defined as the outer product of the centered data matrix, e.g. $S_B = \Sigma_{i=n+1}^{m}(\mathbf{I}_i - \bar{\mathbf{I}}_B)(\mathbf{I}_i - \bar{\mathbf{I}}_B)^\top$, thus, a scattered matrix differs from the sample covariance matrix by only a scatter multiple $\mathcal{S}_B = m \cdot \text{cov}(\mathbf{B})$.

The proof of Lemma 1 is attached in Appendix A.

Let $[(\mathbf{I}_1 - \bar{\mathbf{I}}_A), ..., (\mathbf{I}_n - \bar{\mathbf{I}}_A)]$ be expressed in the short-hand notation of $(\mathbf{A} - \bar{\mathbf{I}}_A)$, and similarly for $(\mathbf{B} - \bar{\mathbf{I}}_B)$ and $(\mathbf{C} - \bar{\mathbf{I}}_C)$. From Lemma 1, it is clear that the SVD of $(\mathbf{C} - \bar{\mathbf{I}}_C)$ equals to the SVD of concatenation of $(\mathbf{A} - \bar{\mathbf{I}}_A)$, $(\mathbf{B} - \bar{\mathbf{I}}_B)$, and the additional vector $\sqrt{\frac{nm}{n+m}}\left(\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_A\right)$, which can be formulated into the following algorithm.

Given $\mathbf{A}$, $\bar{\mathbf{I}}_A$, n and $\mathbf{B}$, from which the SVD of $(\mathbf{A} - \bar{\mathbf{I}}_A)$ is computed to get $\mathbf{U}$ and $\mathbf{\Sigma}$, and the SVD of $(\mathbf{C} - \bar{\mathbf{I}}_C)$ is computed to get $\mathbf{U}'$ and $\mathbf{\Sigma}'$:

1. Compute the mean vectors $\bar{\mathbf{I}}_B = \frac{1}{m}\Sigma_{i=1+n}^{n+m}\mathbf{I}_i$ and $\bar{\mathbf{I}}_C = \frac{n}{n+m}\bar{\mathbf{I}}_A + \frac{m}{n+m}\bar{\mathbf{I}}_B$.
2. Form matrix $\hat{\mathbf{B}} = \begin{bmatrix}(\mathbf{I}_{m+1} - \bar{\mathbf{I}}_B) & \cdots & (\mathbf{I}_{n+m} - \bar{\mathbf{I}}_B) & \sqrt{\frac{nm}{n+m}}(\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_A)\end{bmatrix}$.
3. Compute $\tilde{\mathbf{B}} = \text{orth}\left(\hat{\mathbf{B}} - \mathbf{U}\mathbf{U}^\top\hat{\mathbf{B}}\right)$ and $\mathbf{R} = \begin{bmatrix}\mathbf{\Sigma} & \mathbf{U}^\top\hat{\mathbf{B}} \\ \mathbf{0} & \tilde{\mathbf{B}}^\top\left(\hat{\mathbf{B}} - \mathbf{U}\mathbf{U}^\top\hat{\mathbf{B}}\right)\end{bmatrix}$, where the operator orth() performs orthogonalization, maybe via QR. (Note that the $\tilde{\mathbf{B}}$ here is one column larger than that in the SKL algorithm.)
4. Compute the SVD $\mathbf{R}$: $\mathbf{R} \overset{\text{SVD}}{=} \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^\top$.
5. $\mathbf{U}' = [\mathbf{U}\ \tilde{\mathbf{B}}]\tilde{\mathbf{U}}$ and $\mathbf{\Sigma}' = \tilde{\mathbf{\Sigma}}$.

Figure 2.4: The incremental PCA algorithm.

The Forgetting Factor

By incorporating what Levey et al. [12] suggested, IVT ensures that the dictionary reflects the more recent target image patches than the earlier ones by implementing the option of forgetting factor f. The forgetting factor is a bias to down-weight the earlier observations so that the newer observations are more significant. The forgetting factor f down-weights the earlier observation and $f \in [0, 1]$, where $f = 1$ indicates no forgetting is to occur. By adding the forgetting factor, $\mathbf{R}$ in step 3 of the incremental PCA algorithm in Figure 2.4 becomes $\mathbf{R} = \begin{bmatrix}f\mathbf{\Sigma} & \mathbf{U}^\top\hat{\mathbf{B}} \\ \mathbf{0} & \tilde{\mathbf{B}}^\top\left(\hat{\mathbf{B}} - \mathbf{U}\mathbf{U}^\top\hat{\mathbf{B}}\right)\end{bmatrix}$, which is equivalent to taking the QR decomposition of $\begin{bmatrix}f\mathbf{U}\mathbf{\Sigma} & \hat{\mathbf{B}}\end{bmatrix}$ instead of $\begin{bmatrix}\mathbf{U}\mathbf{\Sigma} & \hat{\mathbf{B}}\end{bmatrix}$. The incorporation of the forgetting factor is possible according to the following lemma by Ross et al. [18], whose proof is attached in Appendix A:

Lemma 2. A forgetting factor f reduces the contribution of each block of data to the overall covariance modeled by an additional factor of f^2 at each SVD update.■

Therefore, at the k^{th} eigenbasis update, the block of m observations added during the j^{th} update ($j < k$) will have its covariance down-weighted by a factor of $f^{2(k-j)}$.

The forgetting factor on the basis mean is addressed in the similar sense by reducing the covariance contribution and its contribution to the resulting mean. By implementing the forgetting factor, $\bar{\mathbf{I}}_C$ in step 1 of the incremental PCA algorithm in Figure 2.4 is modified to $\bar{\mathbf{I}}_C = \frac{fn}{fn+m}\bar{\mathbf{I}}_A + \frac{m}{fn+m}\bar{\mathbf{I}}_B$.

During the updating process, if the eigenvalue associated with a specific eigenbasis template is less than 10^{-6}, then the template is expelled from the dictionary. This mechanism is to help allowing the dictionary to keep only the relevant eigenbasis templates.

2.1.2 The Dynamic Model: Estimating the Target Position with Particle Filtering

As discussed in section 1.1.3, a warping mechanism can transform the affine parameters to the images patch they represent. A brief discussion of the particle filter mechanism on these parameters is described in section 1.2.1. Let $\mathbf{X}_t = (x_t, y_t, \theta_t, s_t, \alpha_t, \phi_t)$ represent an IVT state $\mathbf{X}$ at time t consisting of six parameters, $x, y, \theta, s, \alpha, \phi$, denoting x translation in pixels, y translation in pixels, rotation angle in radians, scale in fraction, aspect ratio in fraction, and skew angle in radians respectively.

Position estimation is cast as an inference task with Markov assumption in hidden state variables. A set of images observed at time t, $\mathcal{I}_t = \{\mathbf{I}_1, \cdots, \mathbf{I}_1\}$, the values of its hidden state variable $\mathbf{X}$ is estimated using Bayes' theorem

$$p\left(\mathbf{X}_t|\mathcal{I}_t\right) \propto p\left(\mathbf{I}_t|\mathbf{X}_t\right) \int p\left(\mathbf{X}_t|\mathbf{X}_{t-1}\right) p\left(\mathbf{X}_{t-1}|\mathcal{I}_{t-1}\right) d\mathbf{X}_{t-1}. \tag{2.3}$$

This tracking model is governed by the observation model estimating the likelihood of $\mathbf{X}_t$

given the observation $\mathbf{I}_t$, $p(\mathbf{I}_t|\mathbf{X}_t)$, and the dynamic model estimating the likelihood of $\mathbf{X}_t$ given the observation $\mathbf{X}_{t-1}$, $p(\mathbf{X}_t|\mathbf{X}_{t-1})$. The observation model is discussed in detail later in section 2.1.3.

In the dynamic model, each parameter in the state $\mathbf{X}_t$ is modeled with an independent Gaussian distribution around $\mathbf{X}_{t-1}$, each with its corresponding variance. The variances of the parameters are collectively represented by $\mathbf{\Psi} = \left(\sigma_x^2, \sigma_y^2, \sigma_\theta^2, \sigma_s^2, \sigma_\alpha^2, \sigma_\phi^2\right)$. Namely,

$$p(\mathbf{X}_t|\mathbf{X}_{t-1}) = \mathcal{N}(\mathbf{X}; \mathbf{X}_{t-1}, \mathbf{\Psi}), \tag{2.4}$$

where $\mathcal{N}$ denotes Gaussian (or normal) distribution and the search around $\mathbf{X}_{t-1}$ defines the motion of interest to the tracker, i.e. large variance σ_x^2 in x translation means a wider search region in x-direction, hence, a fast movement in x-direction is assumed. The variance for each affine parameter in IVT is assumed to be static, thus, the variances do not change during tracking.

2.1.3 The Observation Model: Estimating the Target Image Patch

The image observations are modeled by a probabilistic interpretation of the principal component analysis (PCA) [28]. A target image patch drawn at time t, $\mathbf{I}_t$, is chosen based on $\mathbf{X}_t$ under the assumption that the chosen target image patch $\mathbf{I}_t$ can be represented by the prior target appearances spanned by $\mathbf{U}$, i.e. dictionary templates, and centered at the mean $\boldsymbol{\mu}$, i.e. eigenbasis template mean. The eigenbasis templates are the culmination of the prior target appearances, and this eigenbasis subspace is centered at the mean of the eigenbasis. The probability of an image patch being generated from this subspace is inversely proportional to the distance d from the image patch to the reference point, i.e. $\boldsymbol{\mu}$. The longer the distance d is from the reference point $\boldsymbol{\mu}$ in the subspace $\mathbf{U}$, the less likely the image patch is relevant and being the target appearance estimation. The distance d can be decomposed into the

distance-to-subspace d_s and the distance-within-subspace from the projected image patch to the subspace center d_w, which is an idea proposed by the analysis of Moghaddam et al. [15].

The probability of an image patch at time t generated from a subspace, denoted by $p_{d_s}(\mathbf{I}_t|\mathbf{X}_t)$, is modeled as a Gaussian distribution by

$$p_{d_s}(\mathbf{I}_t|\mathbf{X}_t) = \mathcal{N}\left(\mathbf{I}_t; \boldsymbol{\mu}, \mathbf{U}\mathbf{U}^\top + \varepsilon\mathbf{I}\right), \tag{2.5}$$

where $\mathbf{I}$ is an identity matrix and the $\varepsilon\mathbf{I}$ term corresponds to the additive Gaussian noise in the observation process. The negative exponential distance from $\mathbf{I}_t$ to the subspace spanned by $\mathbf{U}$, $\exp\left(-\left\|(\mathbf{I}_t - \boldsymbol{\mu}) - \mathbf{U}\mathbf{U}^\top(\mathbf{I}_t - \boldsymbol{\mu})\right\|^2\right)$, i.e. $\exp\left(-\|d_s\|^2\right)$, is proportional to $\mathcal{N}\left(\mathbf{I}_t; \mu, \mathbf{U}\mathbf{U}^\top + \varepsilon\mathbf{I}\right)$ as $\varepsilon \to 0$ [20]. The proof of this is attached in Appendix B.

And within a subspace, the likelihood of the projected image patch can be modeled by the Mahalanobis distance from the mean,

$$p_{d_w}(\mathbf{I}_t|\mathbf{X}_t) = \mathcal{N}\left(\mathbf{I}_t; \boldsymbol{\mu}, \mathbf{U}\boldsymbol{\Sigma}^{-1}\mathbf{U}^\top\right), \tag{2.6}$$

where $\boldsymbol{\Sigma}$ is the matrix of singular values corresponding to the columns of $\mathbf{U}$.

By accounting for the likelihood of outside and within the subspace, the overall likelihood of a image patch being generated from a subspace is

$$p(\mathbf{I}_t|\mathbf{X}_t) = p_{d_s}(\mathbf{I}_t|\mathbf{X}_t)\, p_{d_w}(\mathbf{I}_t|\mathbf{X}_t) = \mathcal{N}\left(\mathbf{I}_t; \boldsymbol{\mu}, \mathbf{U}\mathbf{U}^\top + \varepsilon\mathbf{I}\right)\mathcal{N}\left(\mathbf{I}_t; \boldsymbol{\mu}, \mathbf{U}\boldsymbol{\Sigma}^{-1}\mathbf{U}^\top\right). \tag{2.7}$$

Equation 2.7 is used in IVT for finding the probability of a drawn image patch particle at time t, $\mathbf{I}_t$, to $\mathbf{X}_t$.

Furthermore, as proposed by Black et al. [3], due to the effect of the noisy pixels, the outliers, i.e. the noisy pixels that are unlikely to be part of the target image patch with the current eigenbases, should be ignored. The method proposed to numerically signify the degree of

a particle being an outlier is by using a robust error norm $p(x, \sigma) = \frac{x^2}{\sigma^2 + x^2}$ rather than the common Euclidean norm $\|x\|_2$.

An example of a calculated IVT probability associated with available templates and image patch particles drawn in Appendix D, where the available templates are illustrated by Figure D.1 and the available image patch particles drawn are in Figure D.3. This example is purely for presentation purpose, since the templates in IVT dictionary are stored as eigenbases and would not appear to be easily recognizable object like the ones in Figure D.1. The basis mean, the image patch with the highest probability $p(\mathbf{I}_t|\mathbf{X}_t)$ and the image patch with the lowest probability are shown in Figure 2.5. The basis mean, i.e. the dictionary template mean, is shown to be more similar to the image patch particle with the highest probability than the one with the lowest probability. The image patch particle with the lowest probability, in this example resulted a zero probability, i.e. $p(\mathbf{I}_t|\mathbf{X}_t) = 0$, which means that it has a very high dissimilarity to the basis mean.

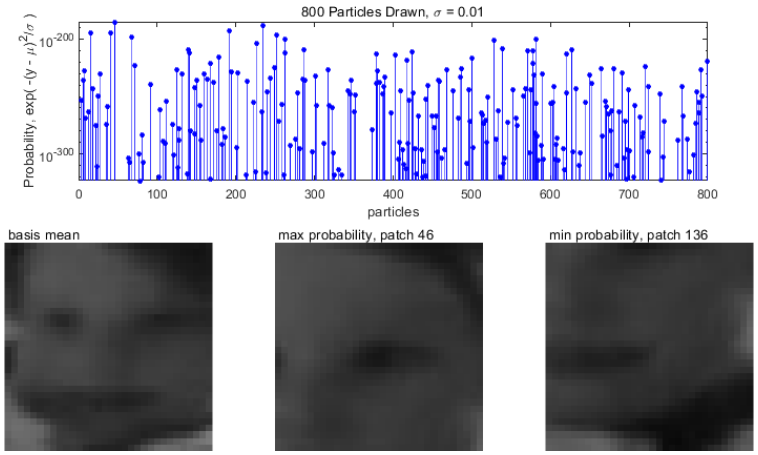


Figure 2.5: An IVT example using of the drawn 800 particles and the available templates in Appendix D. The probability calculated for each particles (top). Bottom images are (from left to right) the basis mean, the highest probable image patch and the lowest probable image patch.

2.2 Least Soft-Threshold Squares Tracking (LSST)

Least soft-threshold squares (LSS) tracking proposed by Wang et al. [30] is a generative tracking method that models the error term with the Gaussian-Laplacian distribution, which uses maximum joint likelihood of parameters to find an LSS distance as a measure of the difference between an observation image patch and the dictionary. As aforementioned in section 2.1.3, care has to be taken with noisy pixels when comparing an image patch with the dictionary. LSS tries to respond to outliers effectively by selecting image patch particles with high similarity to the dictionary, which allows more target representative images to populate the dictionary. Wang et al. propose to pose a linear regression optimization problem assuming Gaussian-Laplacian noises and that being the estimated target image patch alone is not sufficient of it being representative of the target appearance. To address this insufficiency of the estimated target image patch being representative to the target appearance, a reconstruction of the estimated target image patch is necessary before being admitted into the dictionary. This mechanism is elaborated later in this section. The assumption of Gaussian-Laplacian noise is made so that the linear regression can detect outliers in an image patch particle more effectively than detecting noisy pixel using the least-squares distance.

The LSS tracking image patch parameter-to-image transformation scheme is the same as described in section 2.1.

2.2.1 Least Soft-Threshold Squares

The LSS is a linear regression method that assumes that the error vectors adhere to the independent and identically distributed (i.i.d.) Gaussian-Laplacian distribution. It uses a linear model to fit a series of noisy observations by solving the LSS optimization problem

iteratively. The optimal minimal distance achieved serves as a measure of the dissimilarity between the observation vector and the dictionary, in which the larger the resulting distance, the greater the dissimilar the observation has to the dictionary.

Gaussian-Laplacian Noise

A linear model is used to represent how close an image patch is to the linear combination of the dictionary templates. The estimated k-dimensional coefficient vector $\mathbf{x} \in \mathbb{R}^{k\times 1}$ expresses a noisy d-dimensional observation vector $\mathbf{y} \in \mathbb{R}^{d\times 1}$, i.e. the image patch, as a linear combination of the images stored in the dictionary $\mathbf{A}$ that are represented by the columns of $\mathbf{A}$. This is expressed as

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}, \tag{2.8}$$

where $\mathbf{A}$ is the data matrix denoting $\mathbf{A} = [\mathbf{r}_1; \mathbf{r}_2; \cdots; \mathbf{r}_d] \in \mathbb{R}^{d\times k}$, i.e. the eigenbasis matrix (or the dictionary) with the i^{th} row being $\mathbf{r}_i$, and $\mathbf{e}$ is the error (or the residual) term denoting $\mathbf{e} = \mathbf{y} - \mathbf{A}\mathbf{x} = [e_1; e_2; \cdots; e_d]$. Note that $e_i = y_i - \mathbf{r}_i\mathbf{x}$, $i = 1, ..., d$, and $e_1; e_2; \cdots; e_d$ are assumed to be i.i.d. according to some probability density function (PDF). Note that the eigenbasis templates are the columns of the dictionary matrix $\mathbf{A}$, not the rows $\mathbf{r}_i$, and rows are being highlighted here for discussion purpose.

If the image patch particle $\mathbf{y}$ is exactly the same as the $\mathbf{A}\mathbf{x}$ term, then the error term $\mathbf{e}$ would result to be zero, but consider that the image patch particles being drawn as perturbed target image of the previous frame and that the dictionary is a collection of slightly different estimated target images from the past frames, there would certainly exist some dissimilarities.

By maximizing the posteriori probability $p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{x}, \mathbf{y}) \cdot p(\mathbf{y})$, which is the same maximizing the joint likelihood probability $p(\mathbf{x}, \mathbf{y})$, the maximum a posteriori (MAP) estimate of coefficient $\mathbf{x}$ can be obtained. The coefficient $\mathbf{x}$ can be alternatively estimated by

$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} p(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{x}} p(\mathbf{e})$, resulting in maximum likelihood estimation (MLE), with the assumption of a uniform prior.

With the PDF of the errors $f_\theta(e_i)$, the likelihood of the estimator, i.e. the joint probability of the error term $\mathbf{e}$, is $p(\mathbf{e}) = \Pi_{i=1}^{d} f_\theta(e_i)$. Here, θ is the parameter set that characterizes the probability distribution, e.g. Gaussian distribution. And maximizing the likelihood function is the same as minimizing the objective function $L_\theta(e_1, ..., e_d) = \Sigma_{i=1}^{d} p_\theta(e_i)$, where $p_\theta(e_i) = -\log f_\theta(e_i)$.

Assuming Gaussian distributed errors $\mathbf{e}$, where $e_i \in \mathcal{N}(0, \sigma_N^2)$, i.e. e_i being a zero-mean Gaussian random variable with variance σ_N^2 and a PDF $f_{\mathcal{N}}(e_i) = \frac{1}{\sqrt{2\pi}\sigma_N} \exp\left(-\frac{e_i^2}{2\sigma_N^2}\right)$, the MLE solution is the same as the ordinary least squares (OLS) solution

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 . \tag{2.9}$$

Which has the closed form solution $\hat{\mathbf{x}} = \left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top \mathbf{y}$.

If we assume Laplacian distributed errors $\mathbf{e}$ ($e_i \in \mathcal{L}(0, \sigma_L^2)$), i.e. e_i being a zero-mean Gaussian random variable with variance σ_L^2 and a PDF $f_{\mathcal{L}}(e_i) = \frac{1}{\sqrt{2}\sigma_L} \exp\left(-\frac{\sqrt{2}|e_i|}{\sigma_L}\right)$, the MLE solution is the same as the least absolute deviations (LAD) solution

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_1 . \tag{2.10}$$

An OLS is comparatively simpler to solve, but it is sensitive to outliers due to the Gaussian noise assumption, while LAD is comparatively robust to outliers but more difficult to solve as seen in the elaboration that follows.

By using an additive combination of these two components in the error vector $\mathbf{e}$, Equation 2.8 can be alternatively expressed as

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n} + \mathbf{s}, \tag{2.11}$$

with an i.i.d. Gaussian noise vector $\mathbf{n}$ ($n_i \in \mathcal{N}(0, \sigma_N^2)$) and an i.i.d. Laplacian noise vector $\mathbf{s}$ ($s_i \in \mathcal{L}(0, \sigma_L^2)$), thereby, forming the Gaussian-Laplacian distribution [21]. In this application, $\mathbf{n}$ would be able to handle the small dense noise and $\mathbf{s}$ would handle the outliers. The joint PDF is $p(\mathbf{e}) = \Pi_{i=1}^{d} f_{\mathcal{NL}}(e_i)$, and $f_{\mathcal{NL}}(e_i)$ is found by convolution of the two PDFs,

$$\begin{aligned} f_{\mathcal{NL}}(e_i) &= f_{\mathcal{N}}(n_i) * f_{\mathcal{L}}(s_i) \\ &= \int f_{\mathcal{L}}(s_i) f_{\mathcal{N}}(e_i - n_i) ds_i \\ &= \frac{1}{2\sqrt{2}\sigma_N} \exp\left(-\frac{e_i^2}{2\sigma_N^2}\right) \left[\operatorname{erfcx}\left(\frac{\sigma_N}{\sigma_L} - \frac{e_i}{\sqrt{2}\sigma_N}\right) + \operatorname{erfcx}\left(\frac{\sigma_N}{\sigma_L} + \frac{e_i}{\sqrt{2}\sigma_N}\right)\right]. \end{aligned} \tag{2.12}$$

Where $\operatorname{erfcx}(x) = \exp(x^2)\operatorname{erfc}(x)$ and $\operatorname{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t} dt$.

By considering the Laplacian noise term $\mathbf{s}$ as missing values with the same Laplacian prior, the joint likelihood $p(\mathbf{y}, \mathbf{x}, \mathbf{s})$ can be maximized instead, and

$$\begin{aligned} p(\mathbf{y}, \mathbf{x}, \mathbf{s}) &= p(\mathbf{y}|\mathbf{x}, \mathbf{s}) p(\mathbf{x}, \mathbf{s}) \\ &= p(\mathbf{y} - \mathbf{A}\mathbf{x} - \mathbf{s}) p(\mathbf{s}) \\ &= K \cdot \exp\left[-\frac{1}{\sigma_N^2}\left(\frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x} - \mathbf{s}\|_2^2 + \lambda \|\mathbf{s}\|_1\right)\right]. \end{aligned} \tag{2.13}$$

Where $K = \left(\frac{1}{\sqrt{2}\sigma_L}\right)^d \left(\frac{1}{\sqrt{2\pi}\sigma_N}\right)^d$ and $\lambda = \frac{\sqrt{2}\sigma_N^2}{\sigma_L}$. Therefore, to maximize joint likelihood of $p(\mathbf{y}, \mathbf{x}, \mathbf{s})$ is to minimize the function $\frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x} - \mathbf{s}\|_2^2 + \lambda \|\mathbf{s}\|_1$ with respect to $\mathbf{x}$ and $\mathbf{s}$.

Least Soft-Threshold Squares Regression: Likelihood of Particle to the Dictionary

In Equation 2.13, $\frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x} - \mathbf{s}\|_2^2 + \lambda \|\mathbf{s}\|_1$ is considered as the objective function in maximizing the joint likelihood of $p(\mathbf{y}, \mathbf{x}, \mathbf{s})$,

$$L(\mathbf{x}, \mathbf{s}) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x} - \mathbf{s}\|_2^2 + \lambda \|\mathbf{s}\|_1, \tag{2.14}$$

where the optimal solution is $[\hat{\mathbf{x}}, \hat{\mathbf{s}}] = \arg\min_{\mathbf{x},\mathbf{s}} L(\mathbf{x}, \mathbf{s})$. Equation 2.14 is convex but not differentiable everywhere due the ℓ_1 regularization term on $\mathbf{s}$, thus, there is no closed form solution for this optimization problem. To solve for the optimal solutions $[\hat{\mathbf{x}}, \hat{\mathbf{s}}]$, Wang et al. [30] propose an iterative algorithm called the least soft-threshold squares regression, which iterates till convergence by holding one of the two variables constant, i.e. $\mathbf{s}$, and compute for the other one, i.e. $\mathbf{x}$, then use this solution to compute for the first variable held constant. The proposed computations are as followed.

Proposition 1. Given $\hat{\mathbf{s}}$, the optimal $\hat{\mathbf{x}}$ can be computed by the ordinary least squares solution $\hat{\mathbf{x}} = \left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top (\mathbf{y} - \hat{\mathbf{s}})$.

Proposition 2. Given $\hat{\mathbf{x}}$, the optimal $\hat{\mathbf{s}}$ can be computed by a soft-thresholding (or shrinkage) operation $\hat{\mathbf{s}}_j = \mathcal{S}_\lambda \left([\mathbf{y} - \mathbf{A}\hat{\mathbf{x}}]_j \right)$ where $\mathcal{S}_\lambda = \max(|x| - \lambda, 0) \cdot \text{sign}(x)$ and j denotes the iteration count.

Proposition 1 and 2 imply that $L(\mathbf{x}_{i+1}, \mathbf{s}_{i+1}) \leq L(\mathbf{x}_{i+1}, \mathbf{s}_i) \leq L(\mathbf{x}_i, \mathbf{s}_i)$. Additionally, the convergence to a local minimum is guaranteed by the theorem proposed by Tseng et al. [29], which states: If $f(\mathbf{x}) = g(x) + \sum_{i=1}^{n} h_i(x_i)$ with g being convex and differentiable, each h_i being convex and $\mathbf{x} = (x_1, \cdots, x_n)$, then the block-coordinate descent on x_i's converges to a global minimum of f.

Let $\mathbf{P} = \left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top$ which is computed before the algorithm and set a stopping criterion, e.g. the number of iterations or when a difference between objective values meets a pre-set value. The proposed algorithm is as followed.

Input: An observation vector $\mathbf{y}$, a data matrix $\mathbf{A}$, pre-computed $\mathbf{P} = \left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top$, and a small constant λ.

1. Initialize $j = 0$ and $\mathbf{s}_0 = \mathbf{0}$
2. Iterate
3. Obtain $\mathbf{x}_{j+1}$ via $\mathbf{x}_{j+1} = \mathbf{P}(\mathbf{y} - \mathbf{s}_j)$
4. Obtain $\mathbf{s}_{j+1}$ via $\mathbf{s}_{j+1} = \mathcal{S}_\lambda(\mathbf{y} - \mathbf{A}\mathbf{x}_{j+1})$
5. $j \leftarrow j + 1$
6. Until convergence or termination (of the pre-set criterion)

Output: $\hat{\mathbf{x}}$, $\hat{\mathbf{s}}$

Figure 2.6: The least soft-threshold squares regression algorithm.

And since the objective function $L(\hat{\mathbf{x}}, \hat{\mathbf{s}})$ is convex, the solution is the global minimal solution. A MATLAB code of the algorithm in Figure 2.6 is attached in Appendix C.1.

Least Soft-Threshold Squares Distance

When expressing the data matrix as a concatenation of column vectors $\mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_k]$, where $\mathbf{a}_i$ is the eigenbasis vector, the $\mathbf{A}\mathbf{x}$ term is seen as a linear combination of columns of $\mathbf{A}$, $\mathbf{A}\mathbf{x} = x_1\mathbf{a}_1 + \cdots + x_k\mathbf{a}_k$.

The distance d between a noisy observation $\mathbf{y}$ and the subspace $\mathbf{A}$ (i.e. dictionary) is usually defined to be inversely proportional to the maximum joint likelihood with respect to the coefficient $\mathbf{x}$,

$$d(\mathbf{y}; \mathbf{A}) \propto -\log \max_{\mathbf{x}} p(\mathbf{y}, \mathbf{x}). \tag{2.15}$$

Since $-\log \max_{\mathbf{x}} p(\mathbf{y}, \mathbf{x}) = -\log \max_{\mathbf{x}} p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$, and

$$
\begin{aligned}
-\log \max_{\mathbf{x}} p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) &\propto -\log \max_{\mathbf{x}} p(\mathbf{y}|\mathbf{x}) \\
&\propto -\log \max_{\mathbf{x}} \exp\left(-\frac{1}{2}\left\|\mathbf{y}-\mathbf{A}\mathbf{x}\right\|_2^2\right) \\
&\propto \min_{\mathbf{x}} \frac{1}{2}\left\|\mathbf{y}-\mathbf{A}\mathbf{x}\right\|_2^2 .
\end{aligned} \tag{2.16}
$$

Therefore, for OLS method, the distance can be defined as

$$
\begin{aligned}
d_{OLS}\left(\mathbf{y};\mathbf{A}\right) &= \min_{\mathbf{x}} \frac{1}{2}\left\|\mathbf{y}-\mathbf{A}\mathbf{x}\right\|_2^2 \\
&= \frac{1}{2}\left\|\mathbf{y}-\left(\mathbf{A}^\top\mathbf{A}\right)^{-1}\mathbf{A}^\top\mathbf{y}\right\|_2^2 .
\end{aligned} \tag{2.17}
$$

And for LAD method, the distance can be defined as

$$
d_{LAD}\left(\mathbf{y};\mathbf{A}\right) = \min_{\mathbf{x}} \frac{1}{2}\left\|\mathbf{y}-\mathbf{A}\mathbf{x}\right\|_1 . \tag{2.18}
$$

From these distance definitions, LSS method defines the LSS distance to be

$$
d_{LSS}\left(\mathbf{y};\mathbf{A}\right) = \min_{\mathbf{x},\mathbf{s}} \frac{1}{2}\left\|\mathbf{y}-\mathbf{A}\mathbf{x}-\mathbf{s}\right\|_2^2 + \lambda\left\|\mathbf{s}\right\|_1 . \tag{2.19}
$$

2.2.2 Dynamic Model

As discussed in section 2.1.2, IVT assumes a Markov model with hidden state variables for its motion tracking. The same assumption is made by the LSST, that LSST is a Bayesian inference task. The same affine parameter method that can be used to transition affine parameters into an image patch described in section 2.1, and the same particle drawing approach in accordance to Equation 2.4 is also implemented here.

For t target image patch observations as observed vectors $\mathbf{y}_{1:t} = [\mathbf{y}_1, \cdots, \mathbf{y}_t]$, the estimation of the target state variable $\mathbf{x}$ is formulated as the maximum a posteriori estimation

$$
\hat{\mathbf{x}}_t = \arg\max_{\mathbf{x}_t^i} p\left(\mathbf{x}_t^i|\mathbf{y}_{1:t}\right), \tag{2.20}
$$

where $\mathbf{x}_t^i$ is the i^{th} state particle, and the particle with the highest likelihood throughout frames $1 : t$ is collected in $\mathbf{y}_{1:t}$. From these state particles, corresponding image patch particles are extracted, and an image patch particle with the highest similarity to $\mathbf{y}_{1:t}$ is selected as the estimate target patch $\hat{\mathbf{y}}_t$. The corresponding estimate state $\hat{\mathbf{x}}_t$ of frame t is also known now. And by Bayes theorem, $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ can be formulated as

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1})\, p\left(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}\right) \mathbf{x}_{t-1}, \tag{2.21}$$

where $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ describes the transition from the previous frame $t-1$ to current frame t, in the LSS tracking dynamic model. And $p(\mathbf{y}_t|\mathbf{x}_t)$ is the observation model that estimates the likelihood that an image patch particle belongs to the object class, i.e. subspace spanned by the eigenbasis templates in the dictionary.

2.2.3 Observation Model

The same concept as described in section 2.1.3 is applied here, where the PCA subspace is spanned by the column eigenbasis vectors in $\mathbf{U}$ and the subspace is centered at the basis mean $\boldsymbol{\mu}$. With an i.i.d. Gaussian-Laplacian noise

$$\mathbf{y} = \boldsymbol{\mu} + \mathbf{U}\mathbf{z} + \mathbf{n} + \mathbf{s}, \tag{2.22}$$

where $\mathbf{y}$ is the observation vector, i.e. an image patch particle, $\mathbf{z}$ is the coefficients associated with the eigenbasis vectors, and $\mathbf{n}$ and $\mathbf{s}$ are the Gaussian and the Laplacian noise component respectively.

By formulating Equation 2.22 into the form of Equation 2.9 described in section 2.2.1, it would require letting $\bar{\mathbf{y}} = \mathbf{y} - \boldsymbol{\mu}$, which centers the observed image vector $\mathbf{y}$ at the basis mean $\boldsymbol{\mu}$. Following this convention, the distance between observed image vector $\mathbf{y}$ and the subspace $(\mathbf{U}, \boldsymbol{\mu})$ described by Equation 2.19 in this context is now

$$d_{LSS}\left(\mathbf{y};\mathbf{U},\boldsymbol{\mu}\right)=\min_{\mathbf{z},\mathbf{s}}\frac{1}{2}\left\|\bar{\mathbf{y}}-\mathbf{U}\mathbf{z}-\mathbf{s}\right\|_2^2+\lambda\left\|\mathbf{s}\right\|_1. \tag{2.23}$$

With the solution to the optimization problem with q^{th} particle

$$[\hat{\mathbf{z}}^q,\hat{\mathbf{s}}^q]=\arg\min_{\hat{\mathbf{z}}^q,\hat{\mathbf{s}}^q}\frac{1}{2}\left\|\bar{\mathbf{y}}^q-\mathbf{U}\mathbf{z}^q-\mathbf{s}^q\right\|_2^2+\lambda\left\|\mathbf{s}^q\right\|_1 \tag{2.24}$$

from the LLS regression iterative method described in Figure 2.6, the distance can be found by $d_{LSS}\left(\mathbf{y}^q;\mathbf{U},\boldsymbol{\mu}\right)=\min_{\mathbf{z},\mathbf{s}}\frac{1}{2}\left\|\bar{\mathbf{y}}^q-\mathbf{U}\mathbf{z}^q-\mathbf{s}^q\right\|_2^2+\lambda\left\|\mathbf{s}^q\right\|_1$, and the observation likelihood can be computed as

$$p\left(\mathbf{y}^q|\mathbf{x}^q\right)=\exp\left(-\gamma d_{LSS}\left(\mathbf{y}^q;\mathbf{U},\boldsymbol{\mu}\right)\right), \tag{2.25}$$

where γ is a constant controlling the shape of the Gaussian kernel.

2.2.4 Template Update

In addition to the procedures of admitting an estimated target image into the dictionary, one extra step is taken by the LSS tracking. The best image patch particle chosen $\mathbf{y}_o$ may have noise and admitting a noisy image into the dictionary may corrupt the target appearance representation. From the LSS regression method, the observation vector $\mathbf{y}_o=\left[y_o^1;\cdots;y_o^d\right]$, where y_o^i denotes a pixel of $\mathbf{y}_o$ and each corresponds to the Laplacian noise vector $\mathbf{s}_o=\left[s_o^1;\cdots;s_o^d\right]$, where d is the total number of pixels in the image patch resolution. Recall that from section 2.2.1, a Laplacian noise vector $\mathbf{s}$ handles the outlier noises, meaning that if $s_o^i\neq 0$, there is occlusion or some other noise indicated in the pixel s_o^i. Wang et al. [30] propose to replace these Laplacian noises detected with the respective pixels in the basis mean $\boldsymbol{\mu}$,

$$y_r^i=\begin{cases}y_o^i, & s_o^i=0\\ \\ \mu^i, & s_o^i\neq 0\end{cases}, \tag{2.26}$$

where $\mathbf{y}_r = \left[y_r^1; \cdots ; y_r^d\right]$ denotes the reconstructed vector and $\boldsymbol{\mu} = \left[\mu^1; \cdots ; \mu^d\right]$.

The dictionary eigenbasis templates and the basis mean are found with the same procedures as that of IVT and are updated incrementally as described in section 2.1.1.

An example of a calculated LSST probability associated with the available templates and the image patch particles drawn in Appendix D, where the available templates are illustrated by Figure D.1 and the available image patch particles drawn are in Figure D.3. This example is purely for presentation purpose, since the templates in IVT dictionary are stored as eigenbases and would not appear to be easily recognizable object like the ones in Figure D.1. The basis mean, the image patch with the highest probability and the image patch with the lowest probability are shown at the bottom of Figure 2.7. The basis mean, i.e. the dictionary template mean, is shown to be more similar to the image patch particle with the highest probability than the one with the lowest probability. The dictionary template update process of Equation 2.26 is illustrated in Figure 2.8, where the pixels with non-zero error, i.e. $s_o^i \neq 0$, is replaced with the pixels in the basis mean, i.e. μ^i, so that the reconstructed image patch $\mathbf{y}_r$ admitted into the dictionary would not have the erroneous pixels to cause corruption of the target appearance representation. The resulting reconstructed image patch $\mathbf{y}_r$ has high similarity to the basis mean.

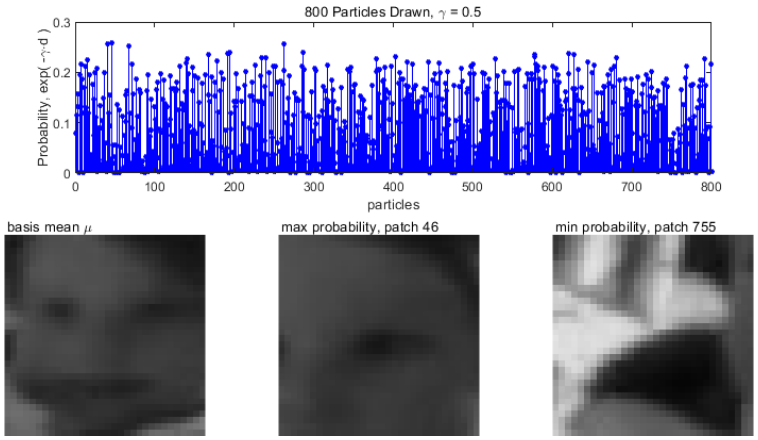


Figure 2.7: An LSST example using of the drawn 800 particles and the available templates in Appendix D. The probability calculated for each particles (top). Bottom images are (from left to right) the basis mean, the highest probable image patch and the lowest probable image patch.

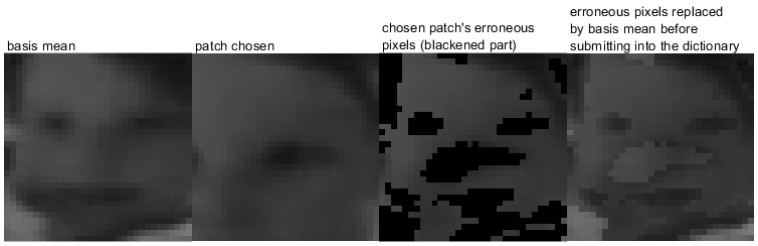


Figure 2.8: An LSST example using of the drawn 800 particles and the available templates in Appendix D to demonstrate the dictionary template update process of Equation 2.26. (From left to right) The basis mean, the chosen image patch particle, the chosen image patch particle with the noisy pixels shown in black, and the reconstructed image patch with the noisy pixels replaced by corresponding pixels in the basis mean before being admitted into the dictionary.

2.3 Robust Visual Tracking (RVT)

Robust visual tracking proposed by Mei et al. [13] solves a one-norm ℓ_1 regularization problem to find the sparse representation composed of three types of templates: target templates, positive trivial templates and negative trivial templates. The positive and negative templates have their corresponding trivial coefficient vectors from the regularization problem, which measure the similarity of the target image particle to the dictionary. The trivial coefficient

vectors indicate outliers in the observed image patch to the dictionary with sparsity constraints, e.g. occlusion, noisy pixels, etc., where the non-zero values in the vectors represent errors in the respective pixels. The dictionary is incrementally updated during the tracking and the templates are populated directly from the most likely image patch particle in the particles drawn. RVT maintains the dictionary templates by accepting estimate target image patches from the video frames which is extracted from the corresponding state variables (affine parameters), and the same affine parameters-to-image patch transformation method described in section 2.1 is used.

RVT maintains the dictionary with image patch template weights. The RVT approach proposes selecting the most likely image patch particle out of the particles drawn, where the particles are drawn based on the particle filtering method discussed in section 2.1.2. Each particle is assigned a weight that indicate how similar the particle is to the dictionary. The more similar the particle is to the dictionary, the heavier the weight is for the respective particle. The most similar particle identified is weighted in the dictionary with respect to its likelihood to the dictionary, i.e. the weight is a function of a particles's similarity to the dictionary. This target image estimate, if accepted by the dictionary, would carry the weight with it into the dictionary as the image patch template weight. The higher the weight in comparison to the other dictionary templates, the less likely the template gets replaced by the future image patch templates. The numerical values of particle likelihood tend to be very small, since there are hundreds of particles drawn. This is seen in Figure 2.5. This method of maintaining the dictionary is to ensure that the best target representation is kept in the dictionary.

A portion of the RVT is omitted in the comparison where RVT dictionary has a set of static templates that do not update incrementally. It is not specified how Mei et al. acquired these static templates, and furthermore, it would require some prior knowledge of the target before tracking begins, which is not the aim in this exploration. Mei et al. choose to have

their image patch in a resolution of 12 pixels-by-15 pixels, total of 180 pixels per image patch, but this will be adjusted to a chosen resolution for all three tracking approaches. Another portion, that is omitted is the part where dynamic affine parameter variances are implemented. This exploration will only use static variances for comparison purpose.

2.3.1 Sparsity Through 1-Norm ℓ_1 Minimization

The target images in the dictionary can be represented in a reduced-dimensional subspace, as mentioned in section 1.1 and the dictionary in IVT and LSST is formed by eigenbases using the method discussed in section 2.1.1. The dictionary in RVT, much like the formulation in section 2.2.1, the dictionary subspace $\mathbf{A}$ is spanned by a set of templates $\mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_k] \in \mathbb{R}^{d \times k}$, where the i^{th} template $\mathbf{a}_i$ is vectorized with d pixels, $\mathbf{a}_i \in \mathbb{R}^d$. The difference is that RVT populates the dictionary directly using the estimated target image patches, this means that $\mathbf{a}_i$ is an image patch accepted by the dictionary and not an eigenbasis template in the eigenbasis subspace. With the coefficient vector $\mathbf{x} = [x_1, \cdots, x_k]^\top \in \mathbb{R}^k$ and the chosen image patch particle in the same resolution $\mathbf{y} \in \mathbb{R}^d$, the chosen image patch can be approximated in the linear span of $\mathbf{A}$,

$$\mathbf{y} \approx \mathbf{A}\mathbf{x} = x_1\mathbf{a}_1 + \cdots + x_k\mathbf{a}_k. \tag{2.27}$$

Similar to the formulation in section 2.2.1, the noise, e.g. occlusion, in the chosen image patch particle can be seen as error $\mathbf{e} \in \mathbb{R}^d$, $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$ as Equation 2.8 showed. And with high similarity between the chosen image patch particle $\mathbf{y}$ and the dictionary $\mathbf{A}$, there should be few low values in the error vector $\mathbf{e}$, where the location of non-zero value elements in $\mathbf{e}$ also indicate the location of the noise.

The error term is formulated differently in RVT. Wright et al. [32] propose to include a collection of trivial templates $\mathbf{I} = [\mathbf{t}_1, \cdots, \mathbf{t}_d] \in \mathbb{R}^{d \times d}$ to capture the occlusions,

$$\mathbf{y} = [\mathbf{A} \quad \mathbf{I}] \begin{bmatrix} \mathbf{x} \\ \\ \mathbf{r} \end{bmatrix}, \tag{2.28}$$

where the $i^{\mathbf{th}}$ trivial template $\mathbf{t}_i \in \mathbb{R}^d$ is made up of all zero entries except for one entry. This means if all trivial templates were to be added together, it would form an identity matrix in the image patch resolution size, in addition, since all the templates are vectorized, $\mathbf{I}$ can be thought of as an identity matrix $\in \mathbb{R}^{d \times d}$. And $\mathbf{r} = [r_1, \cdots, r_d]^\top \in \mathbb{R}^d$ denotes the trivial coefficient vector. And since $[\mathbf{A} \quad \mathbf{I}] \begin{bmatrix} \mathbf{x} \\ \mathbf{r} \end{bmatrix} = \mathbf{A}\mathbf{x} + \mathbf{I}\mathbf{r}$, the $\mathbf{I}\mathbf{r}$ term expresses the error term $\mathbf{e}$ in Equation 2.8.

Mei et al. [13] argue that a tracking template can almost always be represented by the target templates dominated by non-negative coefficients in $\mathbf{x}$, where a image patch particle that has high similarity to the templates are positively related. For instance, the second frame target image patch is very similar to the first frame target image patch stored as template in the dictionary, therefore, the coefficient entries in $\mathbf{x}$ are positive values. Non-negative coefficients help filtering out what is similar to dictionary templates in the image patch particles that are at reversed intensity patterns, e.g. target in brightness transitions into a shadow. Thereby, non-negative constraints are imposed onto $\mathbf{x}$. However, it is unreasonable to impose such constraints onto the trivial coefficient vector $\mathbf{r}$, since it is associated with the error-term and can be positive or negative. Thus, the trivial coefficient vector $\mathbf{r}$ is formulated into a positive and a negative trivial coefficient vector, $\mathbf{r}^+$ and $\mathbf{r}^-$, that contain positive and negative trivial templates respectively.

$$\mathbf{y} = \underbrace{[\mathbf{A} \quad \mathbf{I} \quad -\mathbf{I}]}_{\mathbf{B}} \underbrace{\begin{bmatrix} \mathbf{x} \\ \mathbf{r}^+ \\ \mathbf{r}^- \end{bmatrix}}_{\mathbf{c}} \hat{=} \mathbf{B}\mathbf{c}, \quad \text{s.t.} \quad \mathbf{c} \succeq 0, \tag{2.29}$$

where $\mathbf{B} = [\mathbf{A} \quad \mathbf{I} \quad -\mathbf{I}] \in \mathbb{R}^{d \times (k+2d)}$ and, as the component-wise inequality denoted by $\succeq$,

$\mathbf{c}$ is the non-negative coefficient vector $\mathbf{c} = \begin{bmatrix} \mathbf{x} \\ \mathbf{r}^+ \\ \mathbf{r}^- \end{bmatrix} \in \mathbb{R}^{k+2d}$ with the positive trivial coefficient vector $\mathbf{r}^+$ and the negative trivial coefficient vector $\mathbf{r}^-$.

Since Equation 2.29 is underdetermined, it does not result in a unique solution for $\mathbf{c}$. When there are noises in an image patch particle, only a fraction of the entries in the trivial coefficient vectors $\mathbf{r}^+$ and $\mathbf{r}^-$ reflects the noises. Therefore, a sparse solution to Equation 2.29 is desired. A sparse solution would provide compression in the transform domain by the formulating the one-norm ℓ_1 regularized least-squares problem,

$$\min \|\mathbf{Bc} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{c}\|_1, \text{s.t.} \quad \mathbf{c} \succeq 0, \tag{2.30}$$

where $\|\cdot\|_1$ and $\|\cdot\|_2$ are the one-norm ℓ_1 and two-norm ℓ_2 respectively.

Equation 2.30 is solved by an approach proposed by Kim et al. [6], which solves the problem with an interior-point method. This method uses the pre-conditioned conjugate gradients (PCG) algorithm to compute the search direction, and the amount of time to run the PCG algorithm is dependent on the number of iterations and the cost of a PCG step. The regularization parameter λ determines the number of PCG iterations required by the truncated Newton interior-point method. Admittedly, this problem formulation requires a computationally expensive step in PCG, which is a matrix-vector product that has $O(d(2d+k)) = O(d^2+dk)$, where k is the number of target templates. Mei et al. propose to solve this minimization problem by the routine Kim et al. proposed [7]. Here, a coordinate descent approach similar to the one employed in the LSST method is used. The MATLAB code is attached in section C.1.

The image patch particle that has the smallest residual from projecting onto the target template subspace is the chosen particle. The chosen image patch particle $\hat{\mathbf{y}}$ out of n_p particles drawn $\mathcal{Y} = \left\{\mathbf{y}_t^{(1)}, \cdots, \mathbf{y}_t^{(n_p)}\right\}$ at time t (or frame t), is governed by

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{Y}} \prod_{j=1}^{d} \mathcal{N}\left((\mathbf{y}-\mathbf{A}\mathbf{x})^{(j)}; 0, \sigma^2\right), \tag{2.31}$$

where $\mathbf{x}$ is found by the one-norm ℓ_1 minimization of Equation 2.30 and $(\mathbf{y}-\mathbf{A}\mathbf{x})^{(m)}$ denotes the m^{th} entry in $\mathbf{y}-\mathbf{A}\mathbf{x}$, i.e. the pixel difference. Note that each member in the set $\mathcal{Y}$ has a corresponding state $\mathbf{x}$, and all states belong to the parameter particle set $\mathcal{X} = \left\{\mathbf{x}_t^{(1)}, \cdots, \mathbf{x}_t^{(n_p)}\right\}$.

Importance Weights Associated with Each Particle Drawn

One-norm ℓ_1 minimization favors templates with large norms due to the regularization of $\|\mathbf{c}\|_1$ term. The larger the norm associated with the i^{th} template $\mathbf{a}_i$ is, the smaller the coefficient x_i needed in the approximation of $\|\mathbf{y}-\mathbf{B}\mathbf{c}\|_2$. And since it can be thought of as a numeric indication of the similarity between the image patch $\mathbf{y}$ to the $\mathbf{B}\mathbf{c}$ term that contains the dictionary, Mei et al. propose to define the importance weight w_i to the template $\mathbf{a}_i$ as $\|\mathbf{a}_i\|_2$. Therefore, the larger the weight associated with a template, the more important the template is in the dictionary. Since the templates in the dictionary is updated during the tracking, the weight associated with each template is also updated.

2.3.2 Dynamic Model

Mei et al. specify the affine parameters governing the location and the size of the target patch to be a set of eight affine parameters, $(\alpha_1, \alpha_2, \alpha_3, \alpha_4, t_1, t_2, v_1, v_2)$, where $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ denote patch deformation parameters, (t_1, t_2) denote patch translation parameters in the frame (i.e. pixel translations in x and y), and (v_1, v_2) are the velocity in x and y translations (t_1, t_2). For the purpose of comparing with IVT and LSST, these affine parameters are changed to using the parameters used in IVT and LSST instead. The parameters used in IVT and LSST are described in section 2.1.2. This means that the velocities are not taken into account in RVt,

but the result would offer a one-to-one comparison of IVT, LSS and RVT methodologies.

The particle filter follows the same method with Markov hidden parameters as discussed in section 2.1.2 and expressed in Equation 2.21. With $\mathbf{x}_t$ being the state composed of the image patch affine parameters (i.e. the state variables) at time t, and all the available observations $\mathbf{y}_{1:t-1} = \left\{\mathbf{y}_1, \cdots, \mathbf{y}_{t-1}\right\}$ (i.e. a collection of chosen image patches prior to time t), the predicted distribution up to time $t-1$ is $p\left(\mathbf{x}_t|\mathbf{y}_{1:t-1}\right)$, where

$$p\left(\mathbf{x}_t|\mathbf{y}_{1:t-1}\right) = \int p\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right) p\left(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}\right) d\mathbf{x}_{t-1}. \tag{2.32}$$

In this prediction step, the state vector $\mathbf{x}_t$ is estimated at time t, which the follows updating step of the observation $\mathbf{y}_t$ can be found using Bayes rule

$$p\left(\mathbf{x}_t|\mathbf{y}_{1:t}\right) = \frac{p\left(\mathbf{y}_t|\mathbf{x}_t\right) p\left(\mathbf{x}_t|\mathbf{y}_{1:t-1}\right)}{p\left(\mathbf{y}_t|\mathbf{y}_{1:t-1}\right)}, \tag{2.33}$$

where $p\left(\mathbf{y}_t|\mathbf{x}_t\right)$ is the likelihood. The posterior $p\left(\mathbf{x}_t|\mathbf{y}_{1:t-1}\right)$ is approximated by n_p particles drawn $\mathcal{X} = \{\mathbf{x}_t^q\}_{q=1,\cdots,n_p}$, each member in $\mathcal{X}$ has an importance weight w_t^q associated with it.

2.3.3 Observation Model

The particles drawn at time t are in a set $\mathcal{X} = \{\mathbf{x}_t^q\}_{q=1,\cdots,n_p}$ and the importance distribution in Equation 2.33 denoted by $p\left(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{y}_t\right)$ governs which of the particles in $\mathcal{X}$ to be chosen. The weights of the particles are updated as

$$w_t^q = w_{t-1}^q \frac{p\left(\mathbf{y}_t|\mathbf{x}_t^q\right) p\left(\mathbf{x}_t^q|\mathbf{x}_{t-1}^q\right)}{p\left(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{y}_t\right)}, \tag{2.34}$$

where $\mathbf{y}_t$ is the image patch corresponding to the state particles drawn $\mathbf{x}_t^q$.

And by using the bootstrap particle filter method, $p\left(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{y}_t\right) = p\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right)$. Thus, the

weights become the observation likelihood $p(\mathbf{y}_t|\mathbf{x}_t)$, i.e. the similarity between an image patch to the templates in the dictionary.

The probability of each image patch particle $\mathbf{y}_t$ to the estimated $\mathbf{x}_t$ at time t is $p(\mathbf{y}_t|\mathbf{x}_t)$, which is computed from the error approximation by the dictionary using the one-norm ℓ_1 minimization described in section 2.3.1, specifically, it is described by Equation 2.31 as

$$p(\mathbf{y}_t|\mathbf{x}_t) = \prod_{j=1}^{d} \mathcal{N}\left((\mathbf{y} - \mathbf{A}\mathbf{x})^{(j)}; 0, \sigma^2\right), \tag{2.35}$$

where $\mathbf{A}$ is the dictionary and $\mathbf{x}$ is the coefficient vector, d is the total number of pixels in the image patch of the specified resolution, and σ^2 is the variance in the Gaussian distribution for all pixels in the appearance model. This formulation works under the assumption that a noise in each pixel is an independent Gaussian noise, but in reality, the noises in an image are typically not independent, e.g. when illumination changes, there could be similar noise in the shadow region.

2.3.4 Template Update

Templates are incrementally updated once an image patch is chosen during the tracking. Template update also updates the importance weight associated with the target image patch templates, which is obtained by the zero-mean unit norm of the template.

Mei et al. propose to have ten templates in the dictionary. The dictionary is initialized by populating it with perturbed first target image (which is initialized as the zero-th target image) in four directions (i.e. positive and negative x and y directions) with a small variance $\sigma = 0.0001$ for all directions. Furthermore, the number of templates retained in the dictionary is consistent in all the tracking approaches, i.e. if it is chosen to have ten templates in the dictionary, then IVT, LSS and RVT are all going to retain ten templates in their dictionaries.

Mei et al. propose a template update algorithm that replaces templates, updates templates and updates weights. If the resulted image patch $\mathbf{y}$ has a high dissimilarity to the template set, it will replace the least important template in $\mathbf{A}$ and will then carry the median weight of the current templates. By assigning the median weight to the newly admitted image patch template, the dictionary avoids the newly added templates to be important enough, but not dominating. The weight of each template increases when the similarity between the chosen image patch and the template is high, otherwise the weight decreases.

The similarity function $sim(\cdot, \cdot)$ of RVT is defined as $cos(\theta)$, with θ being the angle between the two normalized input vectors, i.e. the chosen image patch and each templates. The more similar two image patches are, the closer to 1 the similarity function $cos(\theta)$ yields. Mei et al. choose the maximum template weight to be 0.3 to avoid any template being too dominating. Depending on the experiment, a dissimilarity factor τ can be chosen between 0 and 1 to allow admitting a chosen image patch as a new template. The algorithm is as followed.

1. The chosen tracking target $\mathbf{y}_t$, i.e. from the particle set $\mathcal{Y}_t = \left\{\mathbf{y}_t^{(1)}, \cdots, \mathbf{y}_t^{(n_p)}\right\}$ according to the observation model Equation 2.35.
2. The coefficient vector solution $\mathbf{x}$ from Equation 2.30.
3. The current template weights $\mathbf{w}$, such that $w_i \leftarrow \|\mathbf{a}_i\|_2$.
4. A predefined threshold τ.
5. Update weights according to the coefficients of the target templates.
   $w_i \leftarrow w_i * \exp(x_i)$.
   (a) From the largest coefficient $x_m = \max \mathbf{x}$, find the corresponding template $\mathbf{a}_m$.
   $m = \arg \max_{1 \le i \le n_p} x_i$.
6. **if** $(sim(\mathbf{y}, \mathbf{a}_m) < \tau)$
7. $i_0 \leftarrow \arg \min_{1 \le i \le n_p} w_i$
8. $\mathbf{a}_{i_0} \leftarrow \mathbf{y}$ (replacing the existing template with the lowest weight)
9. $w_{i_0} \leftarrow \text{median}(\mathbf{w})$ (replacing the weight of the newly introduced template)
10. **end if**
11. Normalize template weights $\mathbf{w}$ such that $sum(\mathbf{w}) = 1$.
12. Adjust weights $\mathbf{w}$ such that $\max(\mathbf{w}) = 0.3$ to prevent template weight skewing.
13. Normalize templates $\mathbf{a}_i$ such that $\|\mathbf{a}_i\|_2 = w_i$.

Figure 2.9: The algorithm to update templates in RVT.

The template updating algorithm in Figure 2.9 is supposed to reduce the drift of the chosen target patch in the frame. Figure 2.10 illustrates a frame with the estimated target boundary box, the previously chosen image patch and the current target image patch estimate. It also shows all the available templates and the associated weights partly affected by the similarity to the current target image patch estimate.

An example of a calculated RVT probability associated with the available templates and the image patch particles drawn in Appendix D, where the available templates are illustrated by Figure D.1 and the available image patch particles drawn are in Figure D.3. The previously chosen image patch, the currently chosen image patch with the highest probability, i.e. one with the lowest values of coefficients, and the image patch with the lowest probability, i.e. one with the highest values of coefficients, are shown in Figure 2.11.

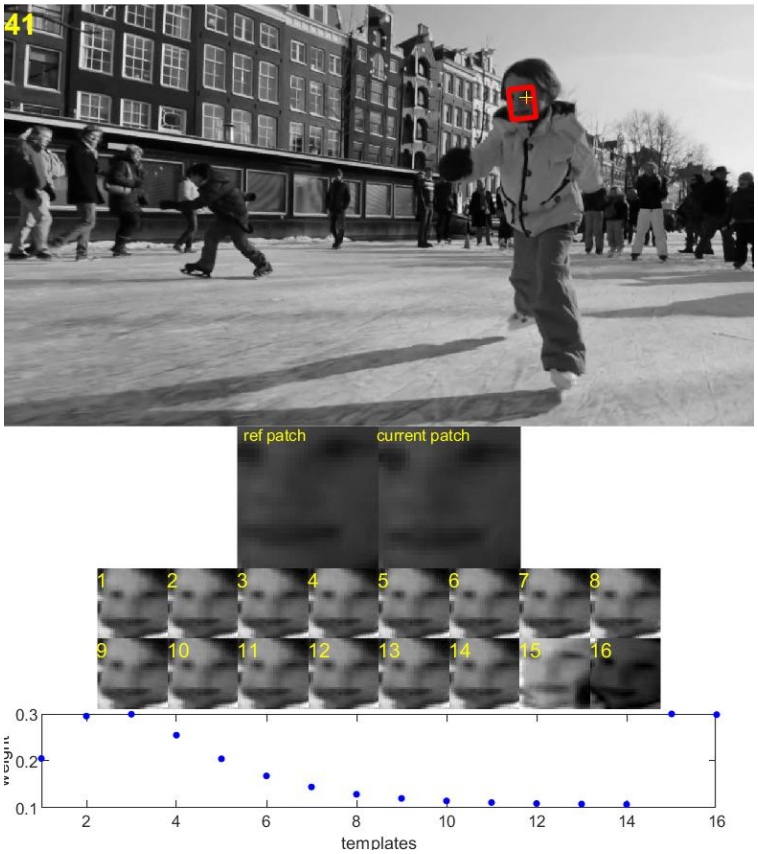


Figure 2.10: An RVT example with the current frame shown with the red estimated target boundary box (top), the chosen image patch from the previous frame and the currently chosen image patch (right beneath the current frame) with all the available templates (right beneath the chosen image patches) and the associated weights (bottom).

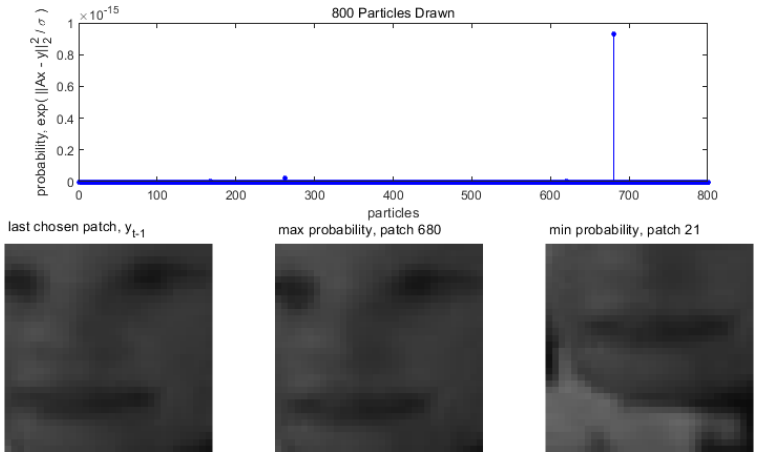


Figure 2.11: An RVT example using of the drawn 800 particles and the available templates in Appendix D. The probability calculated for each particles (top). Bottom images are (from left to right) the chosen image patch from the previous frame, the highest probable image patch and the lowest probable image patch.

2.4 Tracking Approach Comparison

All three tracking approaches, IVT, LSST and RVT, have similar general structures. All three approaches have the same key steps, which are listed in Table 2.1.

1. Specifying the first image patch boundary box on the first frame where affine parameters and image patch in specified resolution are acquired. 2. Drawing particles on the current frame with respect to the previous affine parameters and their variances, where affine parameters are also transformed into image patch particles. 3. A visual tracking method to identify the image patch particle with the highest likelihood, thus, its respective affine parameters. 4. A dictionary updating method and a set of rules to update and acquire new templates in the dictionary.

Table 2.1: The key steps in the general structure that IVT, LSST and RVT have in common.

The structure of RVT slightly varies from IVT and LSST in that between the steps 1 and 2, the whole dictionary is populated using the first image patch with a small x and y translation variance of 0.0001, whereas IVT and LSST populate their dictionary templates on the go.

The methods in which IVT, LSST and RVT handle steps 3 and 4 differ and are the defining characteristics of the tracking approaches, e.g. the "incremental" part of the name in IVT comes from its method to incrementally update the dictionary eigenbasis and the basis mean in step 4.

In step 3, IVT uses the distances between the basis mean-centered image patch particles projected onto the the eigenbasis subspace, and the mean is used as the reference of likelihood the particles being the best tracking estimation, as described by Equation 2.5. LSST also uses distance as a measure of likelihood, but it solves for the distance using a sparse optimization problem of mean-centered particles to the linearly combination of the eigenbases and their errors, posed by Equation 2.14. The more zeros the error vector has for an

image patch, the more likely the distance is small, and the more likely the particle is the target estimate. RVT does not use distance as a measure of an image patch's similarity to the dictionary, but similar to LSST, RVT also solves for a sparse optimization problem. The RVT sparse optimization problem solves for the coefficients with 3 components: one that is associated with the dictionary templates, one associated with positive trivial templates, and one associated with negative trivial templates. The RVT optimization problem is to identify the differences in the particles drawn to the dictionary templates, positive and negative trivial templates, as described by Equation 2.30. The lower the coefficients associated with trivial templates are, the more likely the particle is the best tracking estimation. Note that IVT and LSST assumes that the current target appearance is a linear combination of the eigenbasis templates, whereas RVT assumes that the current target appearance is a linear combination of the past image patches in the dictionary and that RVT does not maintain a dictionary template mean.

Although both LSST and RVT solve for sparse optimization problem, LSST formulates the optimization problem through robust distance (Hubert distance), which is described by the algorithm in Figure 2.6. RVT solves its sparse optimization problem with respect to the dictionary templates and it is much more computationally expensive than solving the LSST sparse optimization problem.

The "weighting" imposed on each dictionary template in IVT and LSST is through updating of the the eigenbasis template via the incremental PCA method as described in Figure 2.4. While RVT assigns the weights of each template partly based on the sparse coefficients associated with the dictionary templates and partly by a set of few rules for a newly admitted template as described by the template update algorithm in Figure 2.9 steps 5 and 9.

In step 4 in Table 2.1, IVT and LSST both follow an incremental PCA method, described by Figure 2.4, to update the dictionary eigenbasis templates and the basis mean. With a target that has highly variable appearances, the basis mean often gets blurry as a result. RVT

updates the dictionary templates with the assumption that the current target appearance is a linear combination of the recent few target image patches preceding it, given that those image patches meet the criterion described by step 6 in Figure 2.9.

The three approaches are summarized in Appendix E with the specifications that are common among the approaches in Table E.1 and that are different in Table E.2 and a summary of the three algorithms in Table E.3.

Chapter 3

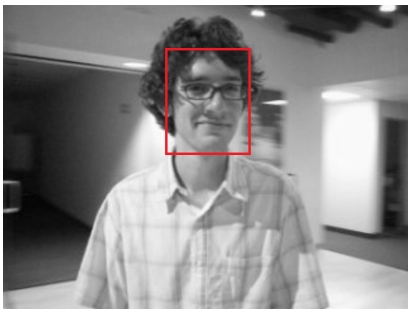
Results

To compare the strength of each tracker, selected representative video samples from Amsterdam Library of Ordinary Videos dataset, called ALOV++ [17], are used. Another source of video samples is from the database set up for the work of "Incremental Learning for Robust Visual Tracking" [18] of Ross et al. [19]. As aforementioned, each video example is converted into gray scale for tracking. The videos selected for the tracking performance examination are recorded at high enough frames rate, usually at 30 frames per second (fps), to avoid blurring of the images in the frame when movement is too large between two frames or when a sudden fast movement creating a large translation variance. These less desirable traits in frames occur in a video called "david_indoor" that has been recorded at 15 fps, where frame images are often blurred as shown in Figure 3.1 and the changing of the movement from frame to frame is sometimes more abrupt than in a 30-fps video. In the tracking performance examination, the videos are chosen to be about 30 fps for consistency.

For performance comparisons of the three tracking approaches, for consistency, 16 templates are kept in the dictionary and 800 particles are drawn at each frame for the dynamic model. But depending on how large the video frame is and its ratio to the size of the target image

Figure 3.1: A frame from "david_indoor" that has blurred image due to being recorded at 15 fps, a low frame rate than 30 fps.

boundary box, the particle drawing variance of each affine parameter is slightly changed for each video. For instance, the video "david_indoor" has a frame size width-by-height 324×240 pixels and a target boundary box size around 80×100 to 50×60 pixels, occupying a third to a fifth of the video frame. Target movement in "david_indoor", therefore, assumed to be less likely to vary than that of the video "01-Light_video00001," which has much higher resolution of 1280×720 pixels and a target image patch size of 65×70 that occupies less than a tenth of the frame. An example frame from each video with target image patch marked is illustrated in Figure 3.2.



(a) A frame from "david_indoor" with a resolution of 324×240 pixels.

(b) A frame from "01-Light_video00001" with a resolution of 1280×720 pixels.

Figure 3.2: Examples of video frame size to the target' boundary box size (marked by the red rectangle).

One initial target boundary box is set for one specific video regardless of the tracking approach. The affine parameters are pixels in x-translation, pixels in y-translation, percentage of image patch scaling, radians of angle rotation, aspect ratio in faction, and skewing in

radians as discussed in section 2.1 and illustrated in Figure 2.2. The target image patch resolution, or the target image patch size, is transformed to be at width-by-height 32×32 pixels when using the dynamic model to identify the next image patch, and the image patches are also stored as such in the dictionary. The higher the resolution of the target image patch, the more detailed and well-represented the image patch is, but also more computationally expensive in the dynamic model. A resolution that is too low may not be detailed enough for identifying good image patch particles. Though, a high resolution of the target image patch may also hinder the ability in tracking, as it contains more details of the target that may not be important in identifying a comparable image patch in the subsequent frames. A slightly lower resolution target image patch is helpful in getting an idea of the appearance without focusing too much on the fine details.

3.1 Quantitative Evaluation

There are a few common comparative methods in measuring tracking performances. These methods are success plot and precision plot from using the one-pass evaluation (OPE) [23] and center location error (CLE) [23]. The evaluations are applied to how the tracking handles certain types of tracking challenges, e.g. occlusion and lighting change.

The precision metric of OPE measures the target boundary box location within a certain threshold distance from the ground truth locations, with the threshold distance pre-set at certain amount of pixels. The success metric of OPE measures the overlapping ratio between the predicted bounding boxes and the ground truth. CLE measures the distance of the center of the bounding box to that of the ground truth.

The OPE and CLE comparison methods are not utilized here, as they require defining the ground truth image boundary box on every frame of the video, which is a long process, and

also, defining the boundary of the target in a frame may be very subjective that varies from one person to another.

One other measure of performance is the root mean square error (RMSE). RMSE measures similarly to CLE, but instead of using the center of the target boundary box to calculate the distance errors, it uses several feature defined on the target to calculate the distance errors. It requires one or more define feature points in the target boundary box and the error distances from those of the estimated image boundary box, then take the square root of the average of the sum of the squared distances. Defining these feature points throughout the video frames is much easier, e.g. the edge of the mouth or one corner of the vehicle. Let d_i be the distance of the i^{th} feature between the ground truth and those of the estimated image patch with a total of n features, where $i = 1, ..., n$,

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} d_i^2}{n}}. \tag{3.1}$$

3.2 Illumination and Scaling

A video named "car4" is a 659-frame video from Ross et al. [18] that is recorded at about 30 fps with the total time length being about 22 seconds. It is selected to examine the tracker's ability to handle illumination change. The apparent target in the video is the rear of a vehicle that goes through illumination changes in the shadow and in the sunlight. These illumination changes are the most prevalent between frames 5 and 10, frames 184 and 187, and frames 231 and 235. The vehicle also goes through a few significant scaling changes. There are 3 ground true features identified for the target in this video.

The initial target boundary box is bounded by the pixel coordinates (247.4259, 82.5498), (247.5498, 77.5741), (242.5741, 277.4502) and (242.4502, 282.4259) using MATLAB's image display convention, where the upper left corner of the image is the origin and x-axis increases

from left to right and y-axis increases from top to bottom. The trackers are set to draw particles with variances of 3 pixels for x-translation, 3 pixels for y-translation, 0.08% scaling, 0.01 radian for rotation, 0.2% for aspect ratio, and 0.001 radian for skew.

3.2.1 IVT

As proposed by Ross et al., IVT is examined with a forgetting factor of 0.95, which retains most of the eigenbasis representation and forgets 0.05 of the previous eigenbasis representations. The dictionary updating frequency is every 5 frames. IVT is able to track the target through illumination and scaling changes, but it struggles a little to track the target with small errors, where at the RMSE increases as the tracking continues, as seen in Figure 3.3. The sudden increase in the RMSE occurs past frame 230, seen in Figure 3.5c, when the target illumination changed, and the tracking started to track a shifted appearance of the target. By around frame 350, seen in Figure 3.5d, the shifted image appearances start to dominate the dictionary and the target boundary box shifts even more off the target till the last frame of the video. Though, despite the shifted target boundary box, the IVT responds well to target scaling throughout the tracking.

During the IVT tracking using the "car4" video, almost every image patch is admitted by the dictionary except for the ones that have eigenvalues lower than 10^{-6}. This is the reason why even an image patch that is less representative of the target appearance, one that has a low non-normalized probability, is admitted into the dictionary. When those image patches that are less representative of the target appearance are admitted into the dictionary, they corrupt the target representations and influence IVT's ability to track in the subsequent frames. This apparent in Figure 3.4, where right before frame 200 when the image boundary box starts to shift, the non-normalized probability of the image to the basis templates becomes really low (low representation of the target appearance). This occurs again around frame 230 and

frame 350 when the target boundary box shifting is worsened. These shifted image patches are accepted into the dictionary and affecting what the dictionary templates represent, as seen in Figure 3.5e.

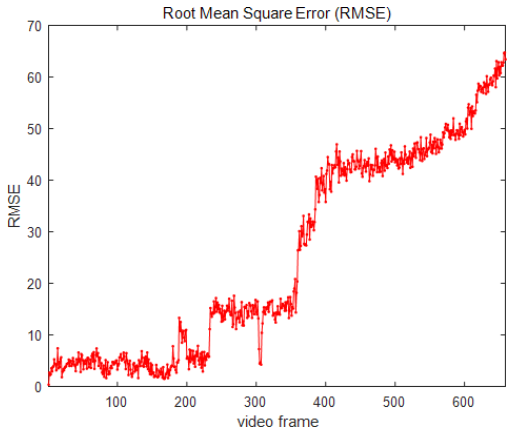


Figure 3.3: The root-mean-square error (RMSE) of the IVT tracking of video "car4" based on the proposal of Ross et al.

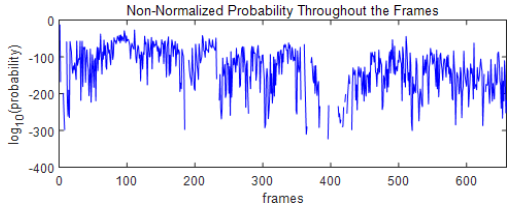
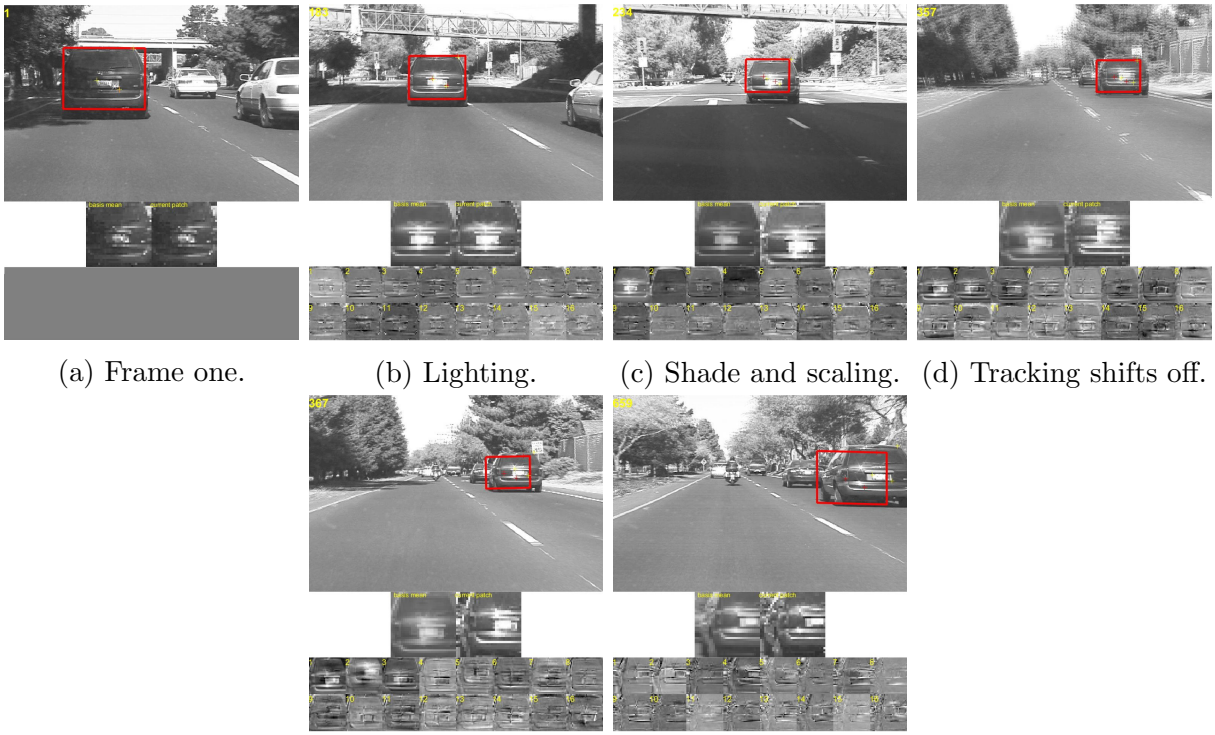


Figure 3.4: The base 10 logarithm of the probability of the chosen image patch to the basis templates from the IVT tracking of video "car4" based on the proposal of Ross et al.



(a) Frame one. (b) Lighting. (c) Shade and scaling. (d) Tracking shifts off.

(e) Dominated by the shifted appearance. (f) The last frame, shifting persists.

Figure 3.5: The key video frames of the IVT tracking performance of video "car4" based on the proposal of Ross et al. The video frame is shown with the target features to be tracked as the ground truth (marked by the yellow crosses), the estimated target feature location (marked by a red cross) and the image boundary box estimate (a red frame). The basis mean is shown below the video frame on the left, while the currently selected image patch is on the right. The dictionary is shown at the bottom, where each template is displayed.

3.2.2 LSST

The optimal solution of Equation 2.23 is computed with λ being 0.5. In this target tracking video, choosing λ value at 0.5 provides adequate optimal solutions. Higher λ values tend to require less iterations to solve the optimization problem, and since 0.5 works well with this video and target, it is implied that any lower values, e.g. 0.1, would result in comparable or better performance. And in alignment to the proposal from Wang et al. [30], the forgetting factor is set to 1 where the existing templates retains all information admitted, and that the dictionary updating frequency is every 5 frames.

LSST is able to track the specified target when encountering target illumination variations with successful results. It is able to grasp the "idea" of the target through the eigenbasis templates, which allows highly probable identifications of the target regardless of illumination and scaling.

A few key frames of the performance are shown in Figure 3.7, and the image patch boundary stays very close to the specified target throughout the video and that it scales itself according to the scaling of the specified target. It can also be seen that the estimated feature points (indicated in red crosses) are very close to that of the ground truth (indicated in yellow crosses), which results in very low RMSEs that stay below 12 pixels as seen in Figure 3.6.

Since LSST and IVT have the same dictionary update algorithm, the difference in performance is entirely from the dynamic model that each utilizes to choose the most probable image patch particle. In the dynamic model, LSST reinforces stronger eigenbasis template representation by "repairing" the part of the chosen image patch that is noisy with that of basis mean, where the dictionary is less likely to be corrupted by the noise. So even if the chosen image patch has a low probability, it is repaired before being admitted into the dictionary, thus, mitigating the issues that IVT presents. This repairing process is illustrated in Figure 2.8. Consequently, LSST does not exhibit the boundary box shift after illumination change and produces very low RMSE to the ground truth, as seen from the stronger target features that the LSST templates have in Figure 3.7.

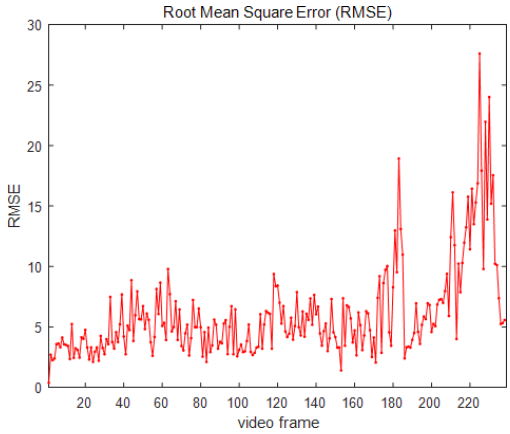
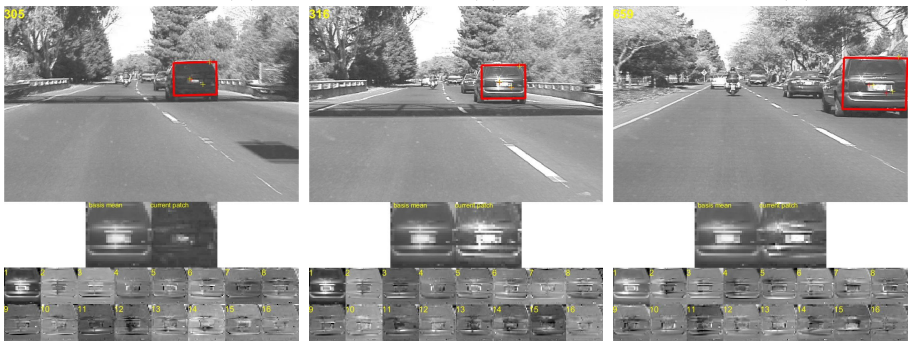


Figure 3.6: The root-mean-square error (RMSE) of the LSS tracking of video "car4" based on the proposal of Wang et al.

(a) Frame one. (b) Lighting. (c) Shade and scaling. (d) Lighting changes.



(e) Lighting changes. (f) Lighting changes. (g) The last frame.

Figure 3.7: The key video frames of the LSS tracking performance of video "car4" based on the proposal of Wang et al. The video frame is shown with the target features to be tracked as the ground truth (marked by the yellow crosses), the estimated target feature location (marked by a red cross) and the image boundary box estimate (a red frame). The basis mean is shown below the video frame on the left, while the currently selected image patch is on the right. The dictionary is shown at the bottom, where each template is displayed.

3.2.3 RVT

RVT is not able to track the specified target when encountering target illumination variations following the proposal by Mei et al. [13]. The issue appears to be that RVT performance is sensitive to illumination changes and that the dictionary update rule hinders template updating. When the illumination changes, RVT chooses an image patch with the least amount of error, but it biases towards the pre-illumination change appearance, which leads RVT to identify an image patch with similar illumination intensity. This error is compounded in the subsequent frames, so that when the target boundary box starts to shift off the target a little bit, it continues the shifting as seen in Figure 3.9b and Figure 3.9c. This is partly due to the dictionary template not being updated during the tracking, where the rule of admitting a new image patch, which is the similarity between the new patch and the heaviest weighted template is below 0.5. But as shown in Figure 3.10, none of the chosen image patches ever has similarity below the value 0.7, as a result, the dictionary never gets updated.

Another reason for RVT's failure to track appears to be that RVT does not truly grasp the "idea" of the target using its dictionary templates. RVT choose the image patch particle just as long as the illumination is similarly represented, which results in RVT failing to identify the image patch of the specified target when encountering illumination changes.

A few key frames of the performance are shown in Figure 3.9, and the image patch boundary initially stays consistent relative to the specified target throughout the video, but during this period, it fails to scale according to the scaling of the specified target. And because the dictionary update rule based on the similarity value "$cos\theta$," as described in section 2.3.4, which hinders dictionary update during the tracking, and RVT only tries to identify an image that has the highest numerical resemblance to those first dark frames. When the templates are made up of bright target image patches and the target encounters a dark illumination change, the scaling of the image patch grows large to capture more dark spots to numerically

satisfy the tracking, which can be seen in Figure 3.9c. These issues are reflected on the RSME illustrated in Figure 3.8, where the error increases during the tracking.

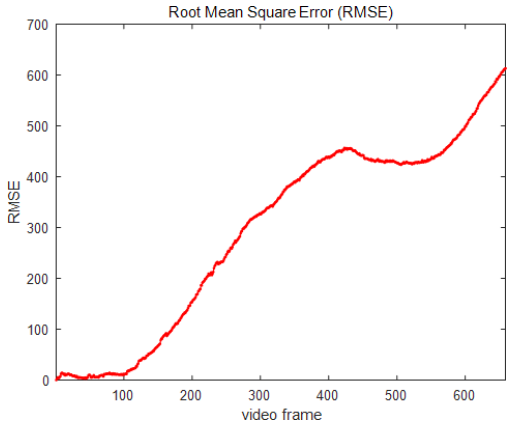


Figure 3.8: The root-mean-square error (RMSE) of the RVT tracking of video "car4" based on the proposal of Mei et al.

(a) Frame one. (b) Patch slightly shifts off the target. (c) Patch lost track of the target.

Figure 3.9: The key video frames of the RVT tracking performance of video "car4" based on the proposal of Mei et al. The video frame is shown with the target features to be tracked as the ground truth (marked by the yellow crosses), the estimated target feature location (marked by a red cross) and the image boundary box estimate (a red frame). The chosen image patch is shown below the video frame on the left, while the currently selected image patch is on the right. The dictionary is shown at the beneath the image patches, where each template is displayed. Below the dictionary is the weighting of each templates.

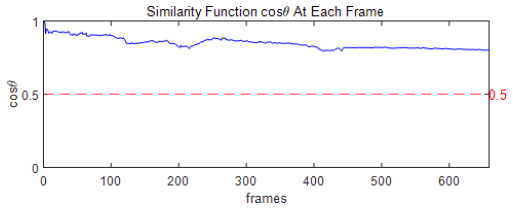


Figure 3.10: The resulting similarity values (blue line) from RVT tracking video "car4" based on the proposal of Mei et al. where the red dash line indicates the criterion of template update (the value of similarity value τ). (An image patch is admitted if similarity value is less than τ.)

3.3 Occlusion and Motion

A video named "01-Light_video00001" from the ALOV++ is selected as an example video to examine how well a tracking performs when presented with temporary target occlusions and varying target motions. This video is composed of 356 frames with a human head as the apparent target, which gets occluded between frames 93 and 97 and once more between frames 133 and 141. There are also partial occlusions and bigger appearance changes, i.e. head turns to the side where a face with all the facial features changes to some features partially visible. The frames prior to the first occlusion allows the trackers to recognize and learn the target appearances, and when the occlusions occur, the tracking performance is examined on how well the trackers respond to the a shorter occlusion of 5 frames and a longer occlusion of 9 frames. The video is recorded at around 30 fps, and the total time length of the video is, therefore, around 12 seconds. Because of the small size of the target boundary box relative to the frame size, there is one ground truth feature identified for the target in this video.

The initial target boundary box is bounded by the pixel coordinates $(1044.7074, 220.7961)$, $(1040.8358, 219.7226)$, $(1035.2926, 259.2039)$ and $(1039.1642, 260.2774)$ using MATLAB's image display convention where the upper left corner of the image is the origin and x-axis

increases from left to right and y-axis increases from top to bottom. The trackers are set to draw particles with variances of 15 pixels for x-translation, 15 pixels for y-translation, 0.08% scaling, 0.01 radian for rotation, 0.2% for aspect ratio, and 0.001 radian for skew.

3.3.1 IVT

As proposed by Ross et al. [18], IVT is examined with a forgetting factor of 0.95, which retains most of the basis representation and forgets 0.05 of the prior information retained. The tracking performs really well before encountering the first occlusion, as shown in Figure 3.13b. Before the first occlusion, the target boundary box is able to scale, rotate and track the target as target appearance scales, rotates and moves around in the frame. The RMSE in Figure 3.11 shows that before the first occlusion, the RMSE value remains below 20. Once past the first occlusion, though IVT tracking boundary box stays in the vicinity of the target, it is unable to identify the target and starts to wander off, as shown in Figure 3.13c. After which, the chosen image patch resembles less and less like the linear combination of the dictionary basis templates as shown in Figure 3.12. Therefore, the chosen image patch shrinks to gain higher probabilistic values, i.e. the average numerical value of the basis templates. The longer the chosen image patch shrinks, the stronger the effect of these near single-color image patches is in the dictionary. This effect can be seen in the dictionary eigenbasis templates of Figure 3.13d, where the templates resemble little of the target appearance, i.e. the features of the target is much softened, and is dominated by the single color appearance.

It can be seen in Figure 3.12 that when the occlusion occurs, the non-normalized probability of the chosen image patch drops steeply around frame 97, which indicate the stark dissimilarity in the image patch. But despite the high dissimilarity of the image patch, this image patch is still admitted by the dictionary and start affecting the basis templates as shown in

Figure 3.13c and Figure 3.13d.

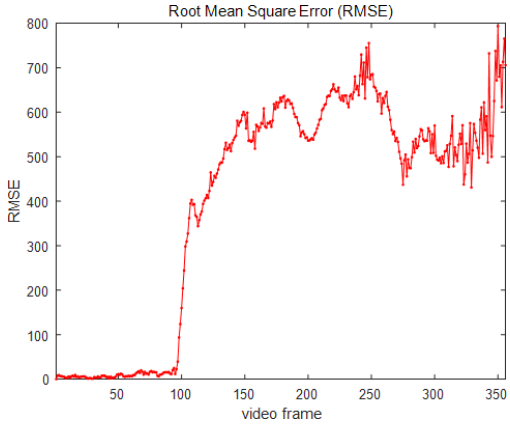


Figure 3.11: The root-mean-square error (RMSE) of the IVT tracking of video "01-Light_video00001".

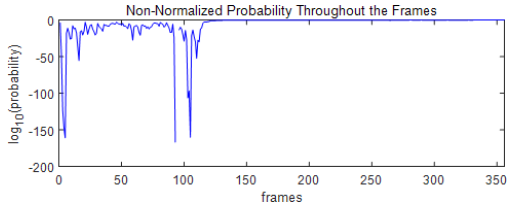
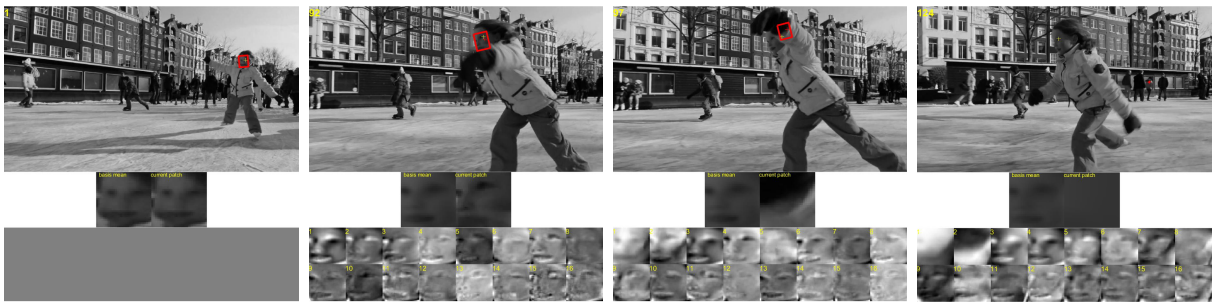


Figure 3.12: The base 10 logorithm of the non-normalized probability in the IVT tracking of video "01-Light_video00001".



(a) Frame one. (b) Before the first occlusion. (c) After the first occlusion. (d) Patch unable to recover and shrinks.

Figure 3.13: The key video frames of the IVT tracking performance of video "01-Light_video00001" based on the proposal of Ross et al. The video frame is shown with the target feature to be tracked (marked by a yellow cross), the estimated target feature location (marked by a red cross) and the image patch (a red frame). The basis mean is shown below the video frame on the left, while the currently selected image patch is on the right. At the bottom is the dictionary, where each template is displayed.

To demonstrate that different tracking approaches may require different settings due to different behaviors, even if tracking using the same video and target, a much more successful tracking performance is presented here. This is done with reduced affine parameter variances of 9 pixels for x-translation, 9 pixels for y-translation, 0.01% scaling, 0.01 radian for rotation, 0.2% for aspect ratio, and 0.001 radian for skew. This set of variances produces similar tracking performance as prior to the variance reduction, where IVT fails to recover from an occlusion in Figure 3.16b. The RMSE also indicates that the IVT lost track of the target before frame 100 in Figure 3.14. A difference with this set of reduced variances is that the target boundary box does not collapse into a smaller area, instead, it stays roughly the same size after losing track of the target, as seen in Figure 3.16c. In Figure 3.15, the non-normalized probability provides an indication where the occlusion occurs with a steep drop, which can be utilized as a measure to detect occluded or noisy image patches.

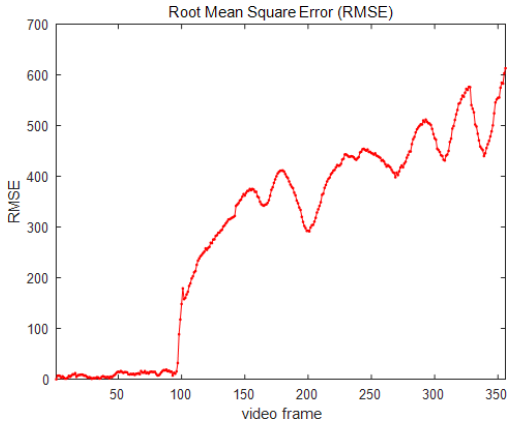


Figure 3.14: The root-mean-square error (RMSE) of the IVT tracking of video "01-Light_video00001" with modified affine parameter variances.

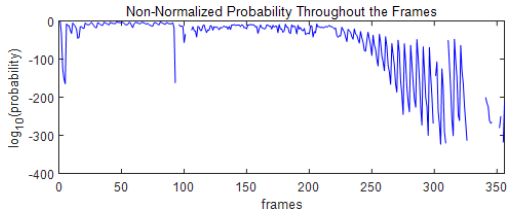
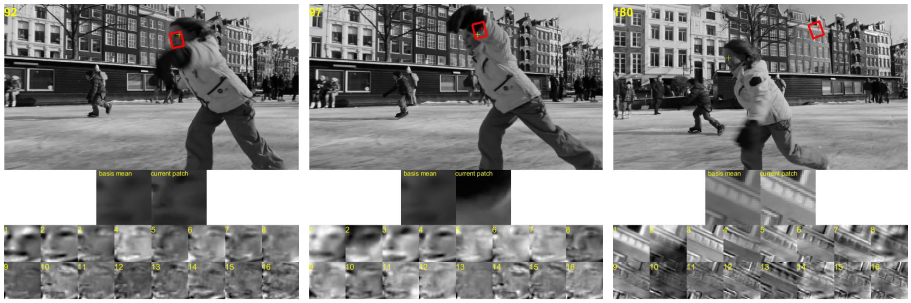


Figure 3.15: The base 10 logorithm of the non-normalized probability in the IVT tracking of video "01-Light_video00001" with modified affine parameter variances.



(a) Before the first occlusion.

(b) After the first occlusion.

(c) Patch unable to recover and but does not shrink.

Figure 3.16: The key video frames of the IVT tracking performance of video "01-Light_video00001" with modified affine parameter variances. The video frame is shown with the target feature to be tracked (marked by a yellow cross), the estimated target feature location (marked by a red cross) and the image patch (a red frame). The basis mean is shown below the video frame on the left, while the currently selected image patch is on the right. At the bottom is the dictionary, where each template is displayed.

3.3.2 LSST

The optimal solution of Equation 2.23 is computed with λ being 0.1. A higher value of λ allows for less iterations and faster results, but it may not produce adequately optimal solutions. In this target tracking video, choosing λ value at 0.1 provides adequate optimal solutions. To align with the proposal from Wang et al. [30], the forgetting factor is set to 1 where the existing templates retains everything, and the dictionary updates every 5 frames.

LSST fails to track a target that goes through significant appearance changes throughout the video, i.e. the full face visible shifting to only one side of the face visible, despite the continual updating of the dictionary. This is a consequence of LSST's inability to differentiate good and bad image patches as the eigenbasis templates try to represent the current and the past image patches. And by accepting almost all image patches into the dictionary, the dictionary fails to preserve a good current target appearance. Once the tracking loses the target, for the image patches to achieve higher probability, the target boundary box tends to become smaller

to capture more single-colored image patches. This phenomenon is analogous to producing an "average" of the eigenbasis templates that produce less errors than capturing a larger region. The size decrease in the target boundary box is significant and abrupt, and once the image patch size decreases too much, the target tracking is unrecoverable and the boundary box will only continue to stay small. The target boundary box shrinking phenomenon is illustrated in Figure 3.18b and Figure 3.18c. This inability for LSST to track the target with significant appearance variations also affects the RMSE, as shown in Figure 3.17, where the error continues to increase during the tracking.

As illustrated in Figure 3.19, the LSST non-normalized probabilities from the chosen image patches are not good indications of whether or not they are similarity to the dictionary templates. This is different from the case in IVT. The minimized distance in Figure 3.20 also does not signal similarity between the chosen image patch to the dictionary templates. This is due to LSST losing track of the target so early in the tracking that the dictionary is corrupted early on.

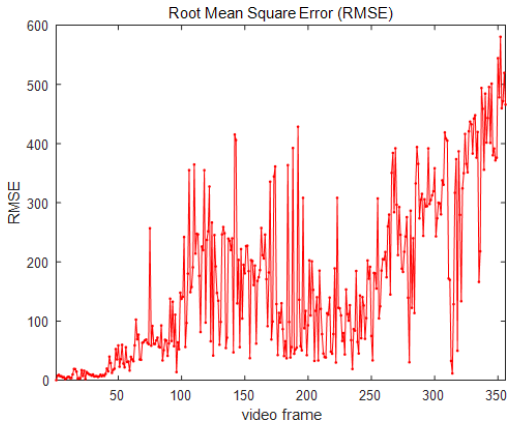


Figure 3.17: The root-mean-square error (RMSE) of the LSST tracking of video "01-Light_video00001".

(a) Frame one. (b) Patch shrinking. (c) Patch unable to recover.

Figure 3.18: The key video frames of the LSST performance of video "01-Light_video00001" based on the proposal of Wang et al. The video frame is shown with the target feature to be tracked (marked by a yellow cross), the estimated target feature location (marked by a red cross) and the image patch (a red frame). The basis mean is shown below the video frame on the left, while the currently selected image patch is on the right. At the bottom is the dictionary, where each template is displayed.

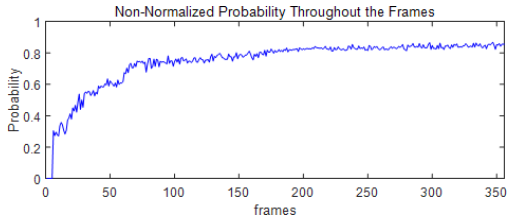


Figure 3.19: The probability of the chosen image patch from the LSST tracking of video "01-Light_video00001".

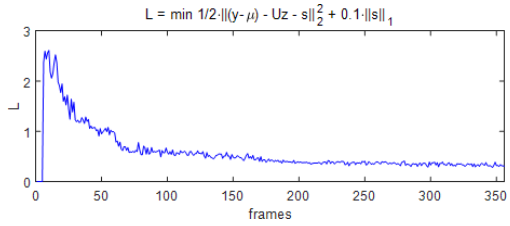


Figure 3.20: The minimized distance of the chosen image patch from the LSST tracking of video "01-Light_video00001".

To demonstrate that LSST has the potential to deal with occlusions and that different tracking approaches may behave differently, which may require different variances even with the same video and target, the affine parameter variances are reduced to 9 pixels for x-translation, 9 pixels for y-translation, 0.01% scaling, 0.01 radian for rotation, 0.2% for aspect

ratio, and 0.001 radian for skew. A rule to reject image patches that are too dissimilar to the dictionary is introduced so that dissimilar image patches do not get admitted by the dictionary. The similarity is defined by the minimized distance d_{LSS}, and if image patches that have resulting d_{LSS} great than 3, then they do not get admitted by the dictionary. This modification in variances allows LSST to track the target closely and recover after occlusions. The RMSE has mostly remains well under 50 pixels, except for the occurrences of the occlusions. The frames illustrating the occlusions are shown Figure 3.22, where the tracking recovers after the occlusions. But partly due to the tracking feature changing after the first occlusion in Figure 3.22b, the target features represented in the eigenbasis templates start to be blurry. And by the end of the tracking, shown in Figure 3.22e, the size of target boundary box is shrunken.

Since in this setting, LSST is able to track the target closely and mainly without the target boundary box shrinking, both the non-normalized probability in Figure 3.23 and the minimized distance in Figure 3.24 indicate where occlusions occur. The non-normalized probability drastically drops when occlusion occur, while the minimized distance clearly spikes up significantly. Either of these values can be used as measures of occlusion occurrence or significant noise detection in the chosen image patches.

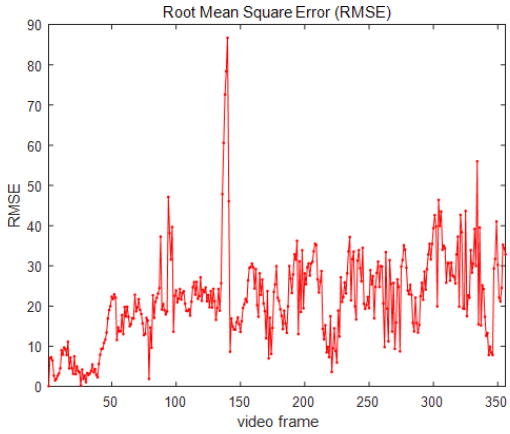
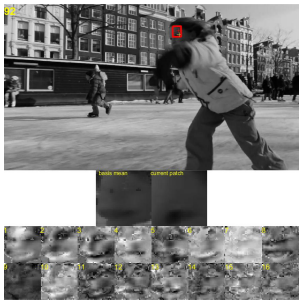
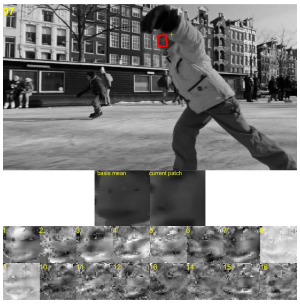


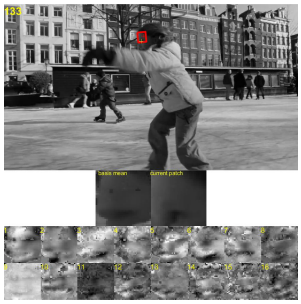
Figure 3.21: The root-mean-square error (RMSE) of the LSST tracking of video "01-Light_video00001" with modified affine parameter variances.



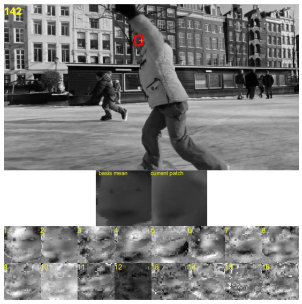
(a) Before the first occlusion.



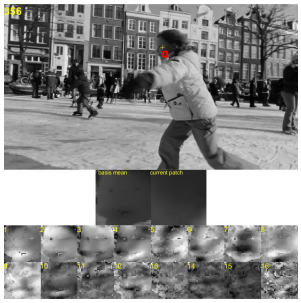
(b) After the first occlusion, tracking a different feature.



(c) Before the second occlusion.



(d) After the second occlusion.



(e) Last frame, patch shrunken.

Figure 3.22: The key video frames of the LSST performance of video "01-Light_video00001" with modified affine parameter variances. The video frame is shown with the target feature to be tracked (marked by a yellow cross), the estimated target feature location (marked by a red cross) and the image patch (a red frame). The basis mean is shown below the video frame on the left, while the currently selected image patch is on the right. At the bottom is the dictionary, where each template is displayed.

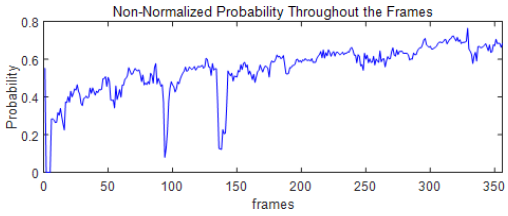


Figure 3.23: The probability of the chosen image patch from the LSST tracking of video "01-Light_video00001" with modified affine parameter variances.

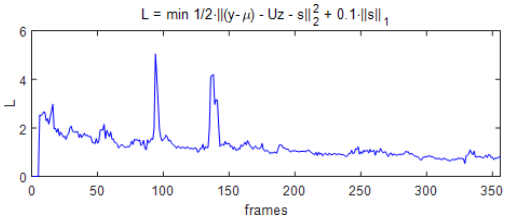


Figure 3.24: The minimized distance of the chosen image patch from the LSST tracking of video "01-Light_video00001" with modified affine parameter variances.

3.3.3 RVT

While RVT can track a target with large appearance changes much better than IVT and LSST, it is unable to recover tracking the target from occlusions with the method proposed by Mei et al. [13], which solves the optimization problem with λ in Equation 2.30 being 0.5. This is seen in Figure 3.26b and Figure 3.26c, where RVT tracks the target very closely before the occlusion, but unable to recover the target after occlusion occurs. And eventually, the image patch shrinks into a dot, as seen in Figure 3.26d, where the chosen image patches can achieve higher probability. This inability to recover from an occlusion may be due to the dictionary not being updated, as shown in Figure 3.26. If the dictionary does not reflect the recent target appearance, RVT cannot identify the image patch with a good target resemblance. Another cause is that RVT identifies the current image patch by referencing to the previous image patch, so that if the previous image patch is corrupted by noises, the current image patch identification is compromised, which can be seen in Figure 3.26c.

The dictionary updating rule is discussed in section 2.3.4, and the similarity value $cos(\theta)$ suggested by Mei et al. is 0.5. With this similarity value, none of the image patches are dissimilar enough to the heaviest weighted template to replace the lowest weighted template. This is illustrated in Figure 3.27, where all the image patches are compared to the heaviest weighted template and none of them is admitted by the dictionary.

The RMSE increases starting from the first occlusion at frame 94, and after which, the tracking is unable to recover the target. This is shown in RSME, where it suddenly spikes up in Figure 3.25.

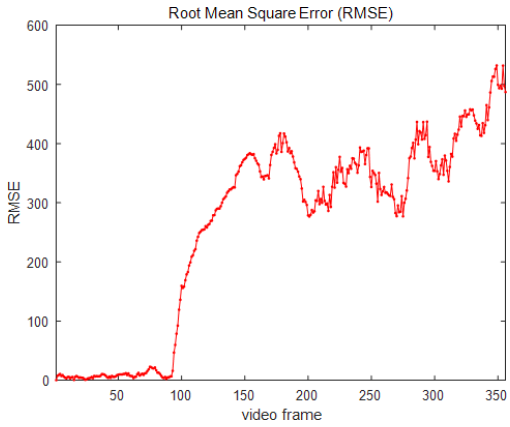
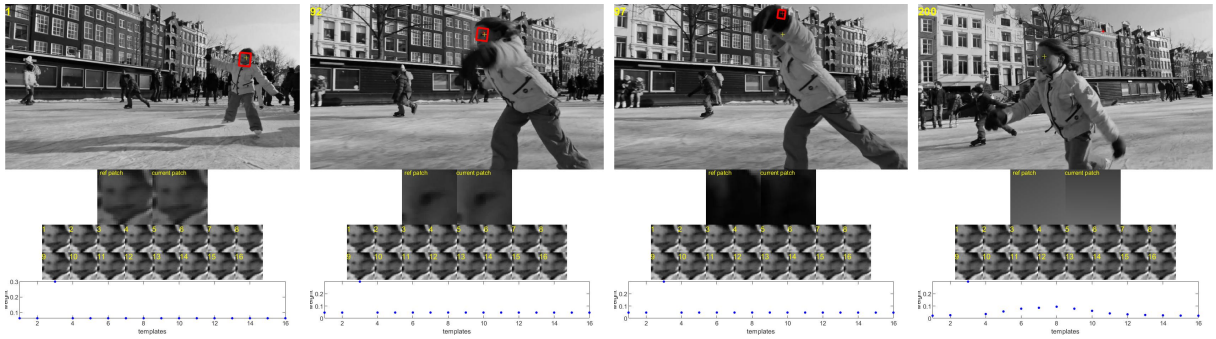


Figure 3.25: The root-mean-square error (RMSE) of the RVT tracking of video "01-Light_video00001".



(a) Frame one. (b) Before an occlusion. (c) After an occlusion. (d) Shrunken image patch.

Figure 3.26: The key video frames of the RVT tracking performance of video "01-Light_video00001" based on the proposal of Mei et al. The video frame is shown with the target feature to be tracked (marked by a yellow cross), the estimated target feature location (marked by a red cross) and the image patch (a red frame). The reference image patch is shown below the video frame on the left, while the currently selected image patch is on the right. Below the image patches are the dictionary templates and beneath the templates is the weighting of each templates.

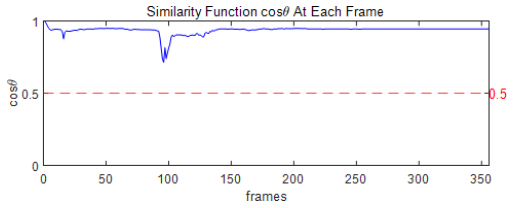


Figure 3.27: The RVT similarity value based on the proposal of Mei et al. of video "01-Light_video00001" for each image patch compared with the heaviest weighted dictionary template at the time, where if the similarity is less then 0.5, the image patch will replace the lowest weighted template and be assigned with medium weight.

3.4 Suggested Modifications Based on the Performance

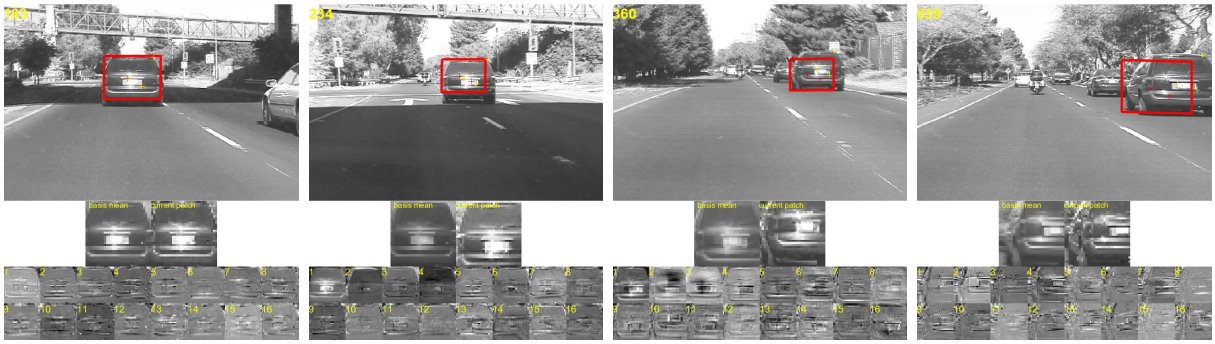
3.4.1 IVT

As discussed by Wang et al. [30] and confirmed by the results presented in the previous sections, the IVT approach demonstrates the ability to handle target appearance changes in illumination, angles and scaling, but it struggles to track when occlusions or when background clutters occur.

The IVT results in section 3.2.1 and 3.3.1 suggest that the IVT dictionary has to be more selective in admitting the chosen image patches. Slight modifications are introduced, while still retain the IVT structure proposed by Ross et al. [18]. A rule is introduced to reject the image patches with low non-normalized probabilities. From studying the probability and failure event occurrences in Figure 3.4 and Figure 3.12 that, with 16 basis templates in the dictionary, a chosen image should not be admitted into the dictionary if the non-normalized probability is below 10^{-50}. The affine parameter variances are chosen to be 9 pixels for x-translation, 9 pixels for y-translation, 0.01% scaling, 0.01 radian for rotation, 0.2% for aspect ratio, and 0.001 radian for skew.

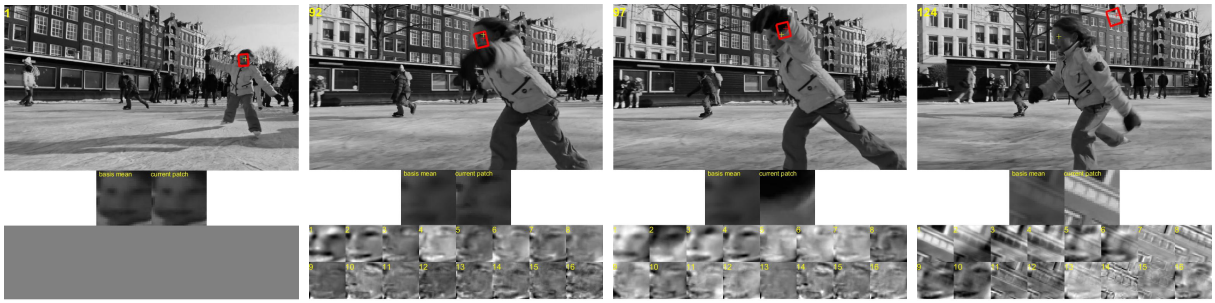
These modifications allow IVT to track the target much more closely after illumination

changes, but they are not sufficient to allow IVT to recover from an occlusion. In the case of tracking the target in video "car4" with these modifications, the target boundary box is less prone to shifting off of target as illustrated in Figure 3.28, where Figure 3.28b shows target boundary box not wandering off after illumination changes and able to respond to target scaling. But shifting off target does eventually occur in Figure 3.28c. In the case of tracking the target in video "01-Light_video00001," Figure 3.29 shows that IVT is unable to recover from an occlusion.



(a) (b) Tracking target closely. (c) Box shifts off target. (d) Last frame, shifting worsens.

Figure 3.28: The key video frames of the modified IVT tracking performance of video "car4." The video frame is shown with the target feature to be tracked (marked by a yellow cross), the estimated target feature location (marked by a red cross) and the image patch (a red frame). The reference image patch is shown below the video frame on the left, while the currently selected image patch is on the right. Below the image patches are the dictionary templates and beneath the templates is the weighting of each templates.



(a) Frame one. (b) Before an occlusion. (c) After an occlusion. (d) A later frame.

Figure 3.29: The key video frames of the modified IVT tracking performance of video "01-Light_video00001." The video frame is shown with the target feature to be tracked (marked by a yellow cross), the estimated target feature location (marked by a red cross) and the image patch (a red frame). The reference image patch is shown below the video frame on the left, while the currently selected image patch is on the right. Below the image patches are the dictionary templates and beneath the templates is the weighting of each templates.

3.4.2 LSST

The ability of LSST to track targets that go through illumination changes is demonstrated in section 3.2.2. From the results in section 3.3.2, LSST struggles to track targets that go through appearance changes. As discussed, one of the reasons is that the dictionary is not representative enough to the more current target appearances. To retain the LSST structure proposed by Mei et al. [13] but still introduce modifications to mitigate the tracking issues, the variances are set at 9 pixels for x-translation, 9 pixels for y-translation, 0.01% scaling, 0.01 radian for rotation, 0.2% for aspect ratio, and 0.001 radian for skew. The results in section 3.3.2 also indicate that the dictionary has to be kept representative to the target appearance. Therefore, in promotion of updating with more target appearance-represented image patches, a rule is introduced to reject any image patches with less than the minimized distance of 3. This image patch rejection rule is similar to the modification described in section 3.4.1.

These modifications provide slight noticeable improvements to LSST's ability to track a

target with relative large appearance changes and occlusions, though RMSE does not appear to have significant changes shown in Figure 3.30. As seen in Figure 3.31, noisy image patches from occlusions are rejected by the dictionary. The target is recovered after the first occlusion in Figure 3.32b, but prior to the second occlusion, the target feature being tracked changes, as seen in Figure 3.32c. The issue with shrinking of the target boundary box is slightly improved in Figure 3.32e.

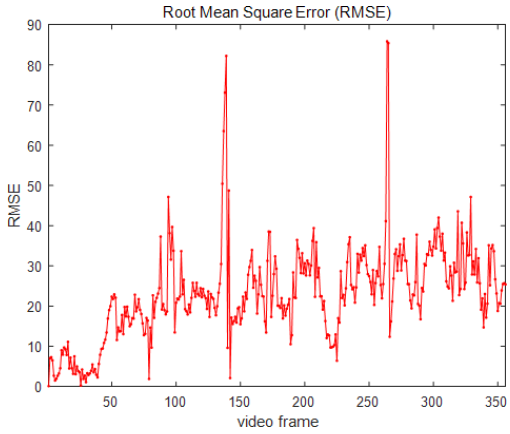


Figure 3.30: The root-mean-square error (RMSE) of the modified LSST tracking of video "01-Light_video00001."

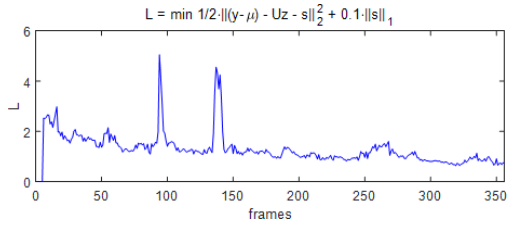
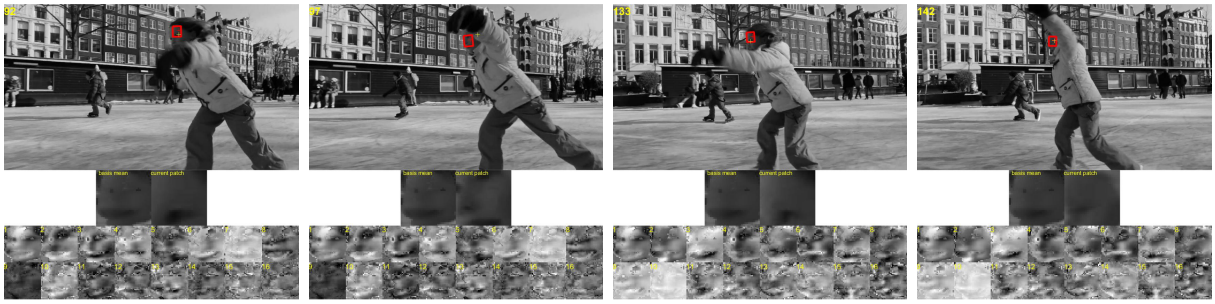


Figure 3.31: The minimized distance of the chosen image patch from the modified LSST tracking of video "01-Light_video00001."

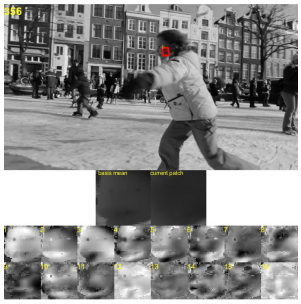


(a) Before the first occlusion.

(b) After the first occlusion, tracking a different feature.

(c) Before the second occlusion.

(d) After the second occlusion.



(e) Last frame, patch shrunken.

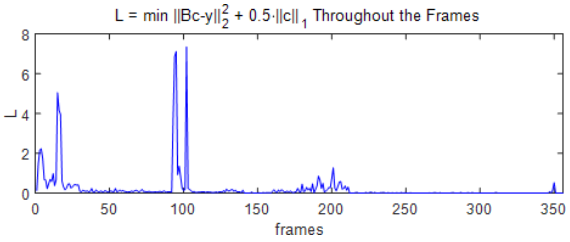
Figure 3.32: The key video frames of the modified LSST performance of video "01-Light_video00001." The video frame is shown with the target feature to be tracked (marked by a yellow cross), the estimated target feature location (marked by a red cross) and the image patch (a red frame). The basis mean is shown below the video frame on the left, while the currently selected image patch is on the right. At the bottom is the dictionary, where each template is displayed.

3.4.3 RVT

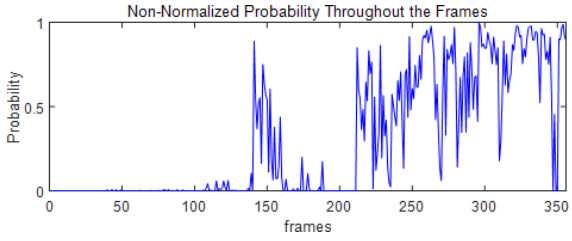
RVT performs well when there is no occlusion present, as it references to the last image patch and the dictionary templates for selecting the current patch. These two items are important factors for selecting the right image patch and for recovery from target boundary box wandering. It is apparent in Figure 3.26 that the dictionary has not been updated throughout the tracking, which means the dictionary is not representative of the recent target appearance and this hinders RVT's ability to recover from occlusions. And in addition, once an occlusion occurs, RVT references to the prior image patch that might be noisy, which

further hinders its ability to choose the best target appearance estimate.

Template updating in step 6 of the template update algorithm in Figure 2.9 can be easily improved by implementing a similarity value that is reflective of image patch similarity, such as the value of 0.95. This encourages the templates to be more representative of the current appearance. This value of high similarity is set with the assumption that the video is captured fast enough, e.g. 30 fps, so that each consecutive frame is similar to the frame preceding it with only subtle variations. Note this value could change if the recorded frame rate is not as high, in which case, the consecutive frame is less similar to the preceding frame. In addition, to ensure the dictionary only admits templates of the good target appearances, a second rule can introduced to not admit any image patch when the dissimilarity is too high. As shown in Figure 3.33a, when occlusion occurs between frames 93 and 97, the minimized objective function Equation 2.30 yields a much higher value due to the high dissimilarity, and the rule introduced can reference this as a measure of how dissimilar the image patch is to the dictionary. The similarity value is modified to be set at 4.5. This new rejection rule is comparatively less effective if it were to be referencing the probability values from Equation 2.33, as shown in Figure 3.33b, where the non-normalized probability being less indicative of the occlusions.



(a) The minimized objective function value of the chosen image patch.



(b) The non-normalized probability of the chosen image patch with the templates.

Figure 3.33: A comparison of RVT measures in an image patch's similarity to the templates of video "01-Light_video00001" based on the proposal of Mei et al.

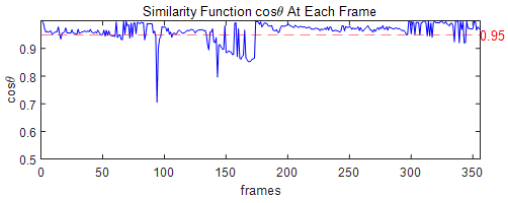
The image patch shrinking phenomenon occurs when there is high dissimilarity in the image patch to the dictionary templates, as illustrated in Figure 3.26d. The shrinkage is due to

RVT's failure to identify an image patch particle that has high enough similarity to the prior images. In an effort to selecting image patches that are more representative of the target appearance when occlusions occur, a new reference method can be introduced so that image patches corrupted by the occlusions is not referenced for selection of the next image patch. The heaviest weighted template is assumed to have the highest similarity to the most recent target appearance and can be referenced when occlusions occur instead of referencing an image patch corrupted by occlusions. But by doing so, the highest weighted template should also be the newest admitted template, therefore, step 9 of the template update algorithm in Figure 2.9 has to be revised so that the newest admitted template is weighted at the maximum weight, and step 12 has to be revised to normalize the weights to between 0 and 0.3. The rule of switching from using the prior image patch as a reference to using the heaviest weighted template is if the minimized objective function from solving Equation 2.30 is larger or equal to 5 so that the image patch is dissimilarly enough to signal that an occlusion or a noisy image patch is detected.

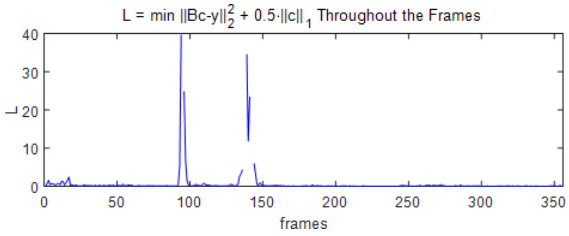
With these modifications, as shown in Figure 3.36, RVT's ability to recover after a temporary occlusion is much improved, and its dictionary templates are much more representative of the target appearance of the moment. Hence, the modifications also improve the RMSE as shown in Figure 3.35. Notice that Figure 3.36d shows that the prior image patch identified is so dissimilar to the dictionary that the heaviest weighted image patch template is referenced to identify the current image patch instead. The frequency of updating the dictionary template can be seen in Figure 3.34b, where when values are below 0.95 and when the minimized objective function does not yield more than 4.5, a new image patch template is admitted. The minimized objective function shown in Figure 3.34a, where the two large peaks are due to the two occlusions in the videos, and because the peaks are higher than 4.5, occlusion corrupted templates being admitted into the dictionary are largely mitigated.

These modifications are not enough to allow RVT to track a target with illumination vari-

ations or target with background clustering, as seen in Figure 3.37 with video "car4." This result compared to that of section 3.2.3 in Figure 3.9 is mildly improved, where the target boundary box size does not immediately change greatly when the illumination changes. But once the tracking loses the target, it is unable to recover the target, unlike the result seen in the case of occlusions.



(a) The similarity values of each image patch.



(b) The minimized objective function values of each image patch.

Figure 3.34: The similarity values and minimized objective function values of video "01-Light_video00001" from running modified RVT.

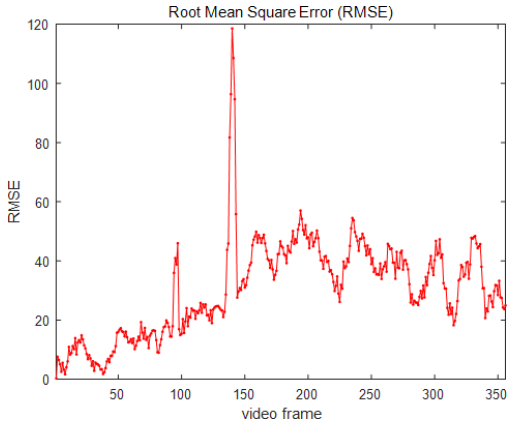
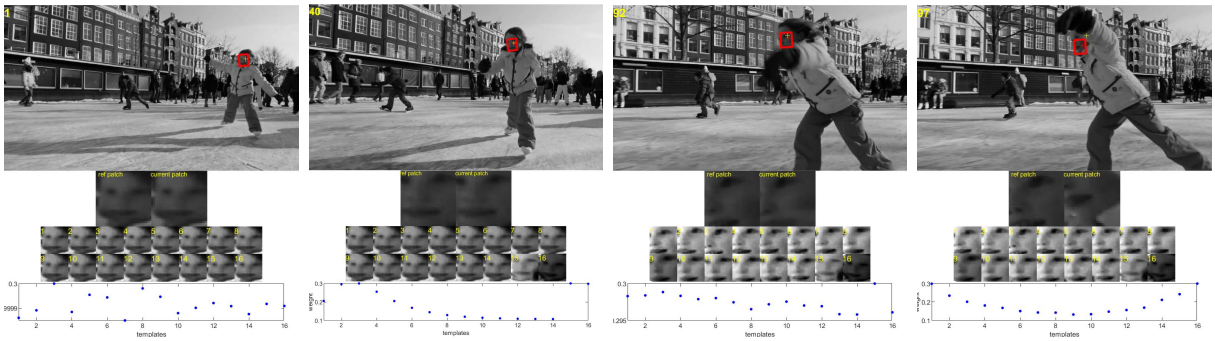


Figure 3.35: The modified RVT root-mean-square error (RMSE) of video "01-Light_video00001."

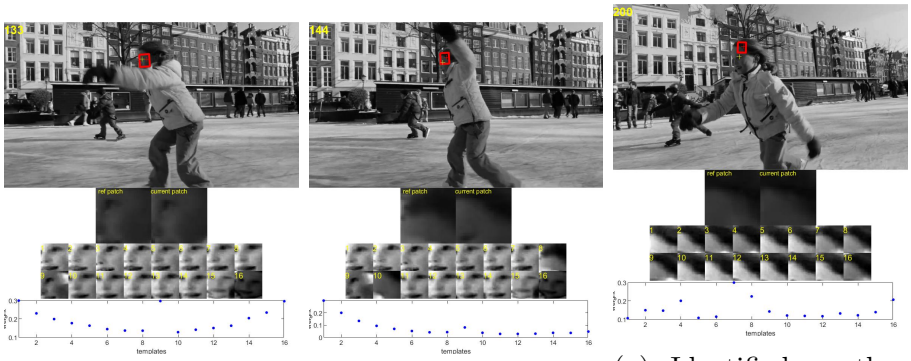


(a) Frame one, dictionary populated.

(b) Dictionary updating while tracking.

(c) Right before the first occlusion.

(d) Right after the first occlusion.

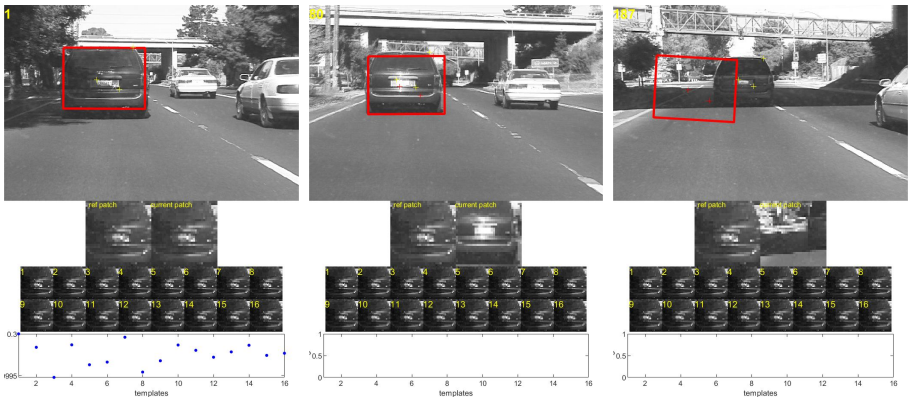


(e) Right before the second occlusion.

(f) Right after the second occlusion.

(g) Identified another feature on the target to track.

Figure 3.36: The key video frames of the modified RVT tracking performance of video "01-Light_video00001". The video frame is shown with the target feature to be tracked (marked by a yellow cross), the estimated target feature location (marked by a red cross) and the image patch (a red frame). The reference image patch is shown below the video frame on the left, while the currently selected image patch is on the right. Below the image patches are the dictionary templates and beneath the templates is the weighting of each templates.



(a) Frame one, dictionary populated.

(b) Target being tracked under similar illumination.

(c) Illumination change causes tracking to lose the target.

Figure 3.37: The key video frames of the modified RVT tracking performance of video "car4". The video frame is shown with the target feature to be tracked (marked by a yellow cross), the estimated target feature location (marked by a red cross) and the image patch (a red frame). The reference image patch is shown below the video frame on the left, while the currently selected image patch is on the right. Below the image patches are the dictionary templates and beneath the templates is the weighting of each templates.

3.4.4 Additional Remarks

It is observed that the tracking approaches IVT, LSST and RVT are sensitive to the initial target boundary box defined. If the initial target boundary is defined slightly differently, though, it is clear what target is being tracked to nominal human recognition, it could result in a large mathematical variation in computation. And this happens if there is no pre-existing concept of the target object and tracker is unable to differentiation of the target and the background.

These approaches also have limited ability in tracking an target that continues to rotate, i.e. a target assumed at position 0° in frame 1 to 30° in frame 150, even if target rotates slowly with a large enough rotation variance and high enough number of particles drawn, the target boundary box is limited in following the rotation. A component may need to be introduced to address this issue.

Chapter 4

Proposed Modifications In the Tracking Components

In close examination of the performances of IVT, LSST and RVT, it is apparent that within their general tracking algorithm structure, dictionary management and the method of choosing the most likely image patch to update the dictionary are vital to robust tracking. The results from LSST and RVT are particularly noteworthy in this regard.

LSST demonstrates exceptional performance when illumination and scaling changes occur, as long as the target features do not vary much. This is due using eigenbasis templates to grasp an "idea" of the target being tracked regardless of the illumination. Therefore, keeping templates representative of the target is vital and a template rejecting mechanism needs be introduced to keep out image patches that are not representative or are not close to the linear combination of the dictionary templates. The disadvantages of LSST are that it has limited ability to track a target with large appearance changes and appears to struggle with occlusions as seen in section 3.3.2 and 3.4.2.

Based on the results in section 3.2.3 and 3.3.3, and the remarks in section 3.4.3, RVT

has a good potential to track targets with high variations in appearance and cope with temporary occlusions, which are all very dependent on the dictionary and dictionary template weights being well-managed. However, RVT struggles to track targets with high illumination variations.

These traits in LSST and RVT appear to be complimentary to each other, where RVT excels at is where LSST struggles in performance and vice versa. These observations inspired the proposal of implementation of weighted particles procedure and keeping two types of dictionaries, where one dictionary keeps eigenbasis templates to provide the tracker the "idea" of the target and the other keeps track of image patches as the templates to allow the tracker to work with large target appearance variations. Keeping two sets of dictionaries is also proposed by Mei et al. [13] on a different ground, where one dictionary keeps the image patches admitted and the other one has pre-collected images of the tracking target for more accurate target identifications. But in order to ensure that the tracking is executed without any prior knowledge of the specific target, there is no pre-collected target images in the dictionary in this proposal.

The MATLAB code of the proposed modifications can be found in Appendix C section C.2. The main structure of the routine is based on the code from Ross et al. [19], where the main routine is in subsection C.2.1 and the subroutines that are either modified or added can be found in section C.2.

4.1 Tracking Routine Structure

The general tracking routine structure of this proposal is aligned with the structure described in section 2.4 and in Table 2.1, but the main components are based on LSST. As in LSST, the observation model and the dynamic model remains the same as described in section 2.2

and an eigenbasis template dictionary is referenced for identifying the current target image patch. This proposed routine implements the template rejection rule introduced in the modified LSST in section 3.3.2 with the allowable d_{LSS} being less or equal to 2.8 and a second set of dictionary that contains the last few admitted image patches in the dictionary. There are 16 eigenbasis dictionary templates and 8 image patch dictionary templates. The criterion for being admitted into the dictionaries is that the image patch must have the distance d_{LSS} in Equation 2.23 be less than 2.8. This rule is due to the smaller distance d_{LSS} is, the more similar the chosen image patch is to the eigenbasis template dictionary.

4.2 Image Patch Template Dictionary

As discussed in section 3.3.2 and 3.4.2, when an occlusion occurs or when handling relative fast changing target appearance, LSST has limited ability to recover. When a chosen image patch has a large distance d_{LSS} with respect to the eigenbasis templates, the probability calculation to reference the last image patch is triggered. The last image patches is assumed to represent a recent target appearance and have stronger features than a linear combination of all eigenbasis templates, therefore, it is a stronger reference to identify the target image patch. The distance d_{LSS} value trigger is set at higher than 2.8, presumably, an occlusion or a big target appearance change has been detected, then the mean $\boldsymbol{\mu}$ and the eigenbasis dictionary $\mathbf{U}$ in Equation 2.23 at time t is replaced with the newest admitted template $\mathbf{a}_k^{U_{IP}}$ where k is the newest template and referencing the image patch dictionary $\mathbf{U}_{IP}$.

4.3 Particle Weights

Another component introduced to the tracking is image patch particle weights based on the area change of the target boundary box. This is introduced to try to mitigate the box

shrinking issue seen in section 3.3.2 and 3.3.2. The particle weights are multiplied with their respective particle probabilities of Equation 2.25, resulting in a weighted probability for the q^{th} particle

$$p\left(\mathbf{y}^q|\mathbf{x}^q\right) = \exp\left(-\gamma d_{LSS}\left(\mathbf{y}^q; \mathbf{U}, \boldsymbol{\mu}\right)\right) \cdot w_{ar}^q, \tag{4.1}$$

where $\mathbf{y}$ is the image patch particle, $\boldsymbol{\mu}$ is the basis mean, $\mathbf{x}$ is the affine parameters of the image patch particle $\mathbf{y}$, γ is a constant between 0 and 1 controlling the shape of the Gaussian kernel, d_{LSS} is calculated using Equation 2.23 referencing the eigenbasis dictionary $\mathbf{U}$, and w_{ar} is the particle weight based on the area change ratio from the previously chosen image patch affine parameters. Note that when the image patch dictionary is used, $\mathbf{y}_{t-1}$ is used in place of $\boldsymbol{\mu}$ and that the image patch dictionary $\mathbf{U}_{IP}$ is used instead,

$$p\left(\mathbf{y}^q|\mathbf{x}^q\right) = \exp\left(-\gamma d_{LSS}\left(\mathbf{y}^q; \mathbf{U}_{IP}, \mathbf{a}_k^{U_{IP}}\right)\right) \cdot w_{ar}^q. \tag{4.2}$$

The image boundary box area change ratio w_{ar} at time t for the q^{th} particle is defined as

$$w_{ar}^q = \frac{1}{1 + \left|\frac{a_{t-1}}{a_t^{(q)}} - 1\right|}, \tag{4.3}$$

where a_{t-1} is the area of target boundary box of the previous estimation at time $t-1$ and $a_t^{(q)}$ is the area of the q^{th} particle boundary box at time t. A computation of the target boundary box is detailed in Appendix F.

4.4 Evaluation

Based on the result in section 3.4.2, it is clear that LSST performance improves with the rejection of image patches that are not representative of the dictionary. The performance of implementations of the template rejection and particle weights in LSST is firstly examined

for the tracker's ability to mitigate target boundary box shrinking, then the performance of the complete implementation with the image patch dictionary is examined.

The examination for LSST is with the video "01-Light_video00001," since any improvement will be much more stark than using video "car4" as LSST has a good overall performance with "car4." The initial target boundary box is bounded by pixel coordinates $(1044.7074, 220.7961)$, $(1040.8358, 219.7226)$, $(1035.2926, 259.2039)$ and $(1039.1642, 260.2774)$ using MATLAB's image display convention where the upper left corner of the image is the origin and x-axis increases from left to right and y-axis increases from top to bottom. The trackers are set to draw 800 particles with affine parameter variances of 9 pixels for x-translation, 9 pixels for y-translation, 0.01% scaling, 0.01 radian for rotation, 0.2% for aspect ratio, and 0.001 radian for skew. The LSS optimization Equation 2.23 is solved with λ being 0.1.

4.4.1 Template Rejection And Particle Weights

The shrinkage of target boundary box does slightly improve with particle weights in comparison to LSST with only template rejections in section 3.4.2, but the improvement is not very noticeable. In fact, when comparing with the resulting RMSE in Figure 4.1, the probability of each chosen image to the existing dictionary in Figure 4.2 and the distance d_{LSS} in Figure 4.3, the two versions of LSST appear to be almost identical. A frame-to-frame comparison in Figure 4.4 also appears to be near identical.

Although the change is not very apparent, the particle weights ensure the images with less change in target boundary box area to be more likely to be chosen, thus, increases the chance of identifying the target again if lost tracking of the target.

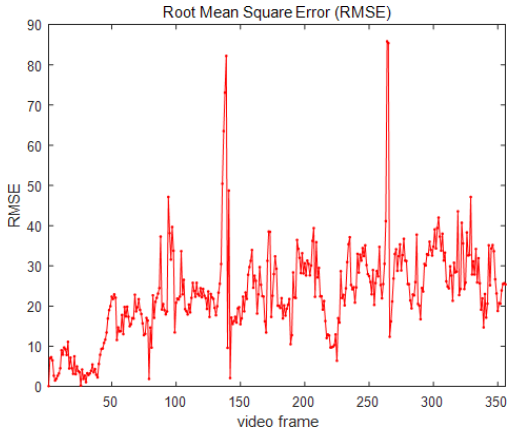


Figure 4.1: The root-mean-square error (RMSE) of the LSST with template rejection and weighted particles tracking of video "01-Light_video00001".

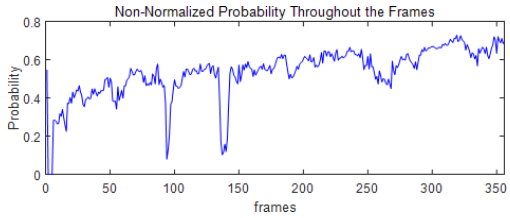


Figure 4.2: The probability of the chosen image patch from the LSST with template rejection and weighted particles tracking of video "01-Light_video00001".

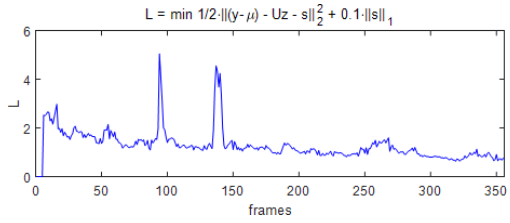
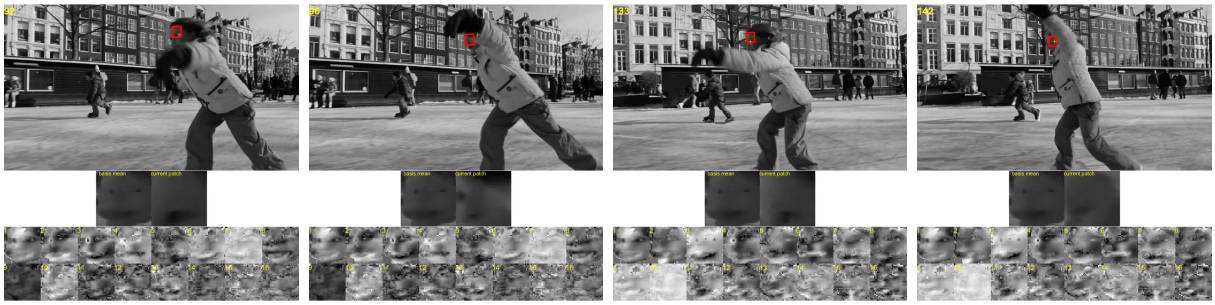


Figure 4.3: The minimized distance of the chosen image patch from the LSST with template rejection and weighted particles tracking of video "01-Light_video00001".



(a) Before the first occlusion.

(b) After the first occlusion, tracking recovers.

(c) Before the second occlusion.

(d) After the second occlusion, tracking recovers.

(e) Some frames later, patch slightly shrunken.

(f) Last frame, patch slightly shrunken.

Figure 4.4: The key video frames of the modified LSST performance of video "01-Light_video00001" with template rejection and weighted particles. The video frame is shown with the target feature to be tracked (marked by a yellow cross), the estimated target feature location (marked by a red cross) and the image patch (a red frame). The basis mean is shown below the video frame on the left, while the currently selected image patch is on the right. At the bottom is the dictionary, where each template is displayed.

4.4.2 All Components Implemented

The result is slightly worse using the image patch dictionary in the case when noise is detected in comparison to LSST performance with only template rejections as discussed in section 3.4.2. This appears to be attributed to the occasional unrepresented templates to be admitted by the dictionaries.

The resulting RMSE in Figure 4.5 is low, but noticeably worse than LSST only implemented with template rejections. The probability of each chosen image patch to the dictionary shown

in Figure 4.6 and the distance d_{LSS} shown in Figure 4.7 also result in less clear divide of noisy image patches and good image patches. In fact, the last frame of the video shown in Figure 4.8f is actually slightly smaller in comparison.

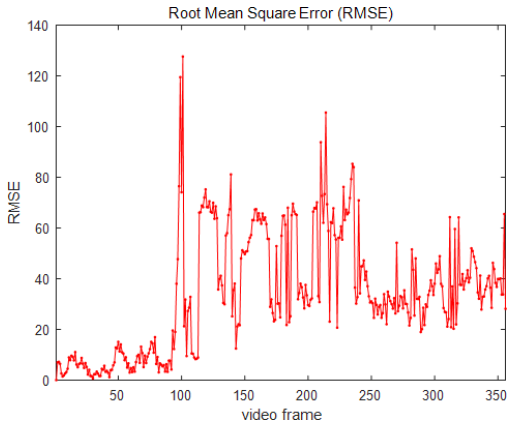


Figure 4.5: The root-mean-square error (RMSE) of the LSST with template rejection and weighted particles tracking of video "01-Light_video00001".

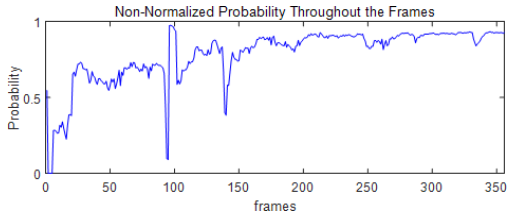


Figure 4.6: The probability of the chosen image patch from the LSST with template rejection and weighted particles tracking of video "01-Light_video00001".

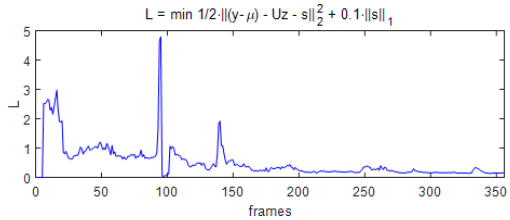
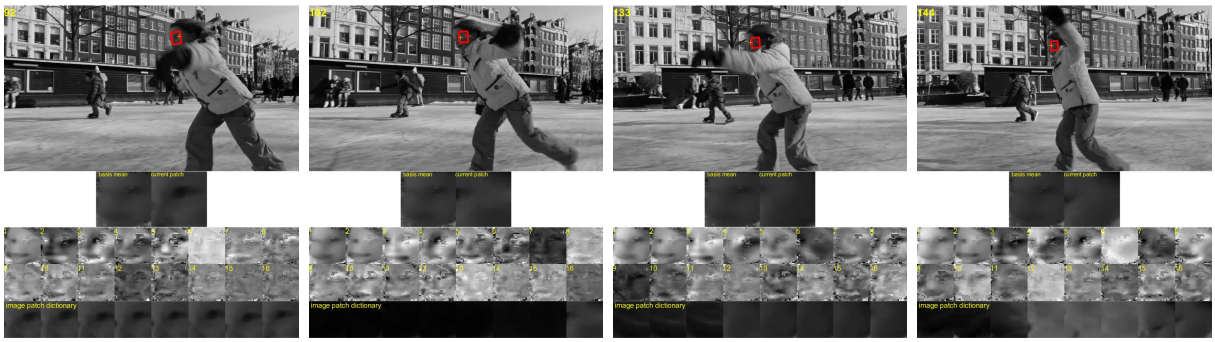


Figure 4.7: The minimized distance of the chosen image patch from the LSST with template rejection and weighted particles tracking of video "01-Light_video00001".

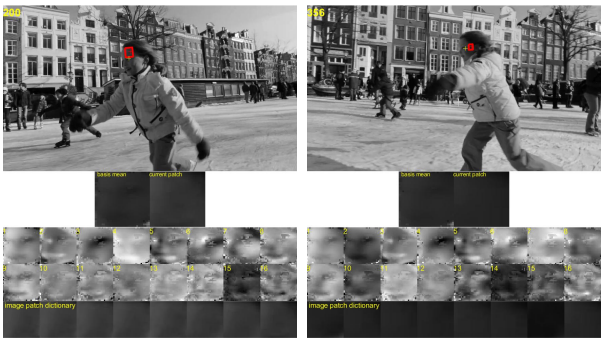


(a) Before the first occlusion.

(b) After the first occlusion, tracking recovers.

(c) Before the second occlusion.

(d) After the second occlusion, tracking recovers.



(e) Some frames later, patch slightly shrunken.

(f) Last frame, patch slightly shrunken.

Figure 4.8: The key video frames of the proposed modified LSST performance of video "01-Light_video00001" with template rejection, weighted particles and two dictionaries. The video frame is shown with the target feature to be tracked (marked by a yellow cross), the estimated target feature location (marked by a red cross) and the image patch (a red frame). The basis mean is shown below the video frame on the left, while the currently selected image patch is on the right. At the bottom is the dictionary, where each template is displayed.

4.4.3 Performance Remarks

The proposal of the components to be implemented in LSST does not yield expected improvement and is slightly worse in some cases. It becomes apparent that maintaining good dictionary references is important, but it is also clear that the definition of "good" image patch with any numerical indication available is difficult. The available numerical indications tend not to have one distinct value that divides the "good" image paches and the "bad."

For instance, using the distance d_{LSS} being higher than 2.8 as a metric of noisy and good image patches is not always a good approach, as seen in Figure 4.7, where many good image patches result in the distance d_{LSS} higher than the value 2.8. And if a lower value, such as $d_{LSS} > 1$ is set as a measure of defining noisy image patches, it would eliminate too many good image patches that are representative of the target appearance. This issue also applies to the probability of each image patch to the existing dictionary, as seen in Figure 4.6. One could implement an average value of change instead, but similar issue persists.

The result of the proposal confirms the limitation of tracking using LSST and similar methods, such as IVT and RVT, and these methods may require additional components that differentiate image patches in different types of computation formulation to provide new measures defining noisy image patches.

Chapter 5

Future Work

Based on the results discussed in chapter 3 and chapter 4, the following elements could be explored to address the limitations or improve the performance in IVT, LSST and RVT.

5.1 Dynamic Affine Parameter Variances

The results presented in chapter 3 are executed with a set of static affine parameter variances. This means that the variances do not change throughout tracking, even if the target starts to speed up its motion.

A version of dynamic affine parameter variances is proposed by Mei et al. [13] in the RVT, where the past five calculated velocities are averaged to estimate new variances for the x and y translations. With updated x and y variances based on the velocity, the search area is more likely to cover the likely search region. The idea for the dynamic variances is the same, but instead of only applying to the variances for x and y translations, it could be applied to all affine parameters, e.g. rotation, scaling, aspect ratio and skew.

If the variances were to change, the number of particles drawn should also be adjusted for frame according to the variances, e.g. based the largest variance change, and perhaps with considerations of the resolution of the video frame and the size of the target boundary box. For instance, if the variances for x and y translations increase, the search area also increases, and therefore, the number of particles would have to increase to capture more image patches in a bigger search area.

The number of velocities to average for the dynamic variances should also consider the frame rate of the video. For instance, in comparison to 30 fps, a 15-fps video would presumably have larger change in target motion velocities and would probably be higher, therefore, the number of velocities to average would be lower than that of a 30-fps video.

5.2 Introduction of A Pre-Trained Tracking Component

Some of the limitations of IVT, LSST and RVT discussed in chapter 3 and especially in section 3.4.4, such as the inablility to differentiate the target object from the background, their limited ability to rotate with the target and the tracking performance being sensitive to the initial target boundary box or the affine parameters, are due to the inability to differentiate one object from the another. A tracking approach with some form of object recognition would mitigate finding a numerically similar image patch without it being representative of the target object.

For instance, a well-trained convolutional neural network (CNN) component could be introduced into the tracking algorithm structure, as CNNs have demonstrated a near-human ability in recognizing objects. Though a CNN requires careful efforts in structure tuning and selecting training data sets as discussed in section 1.1.1. The pre-trained CNN component

could be implemented such that after the image patch particles are drawn, they are fed into the CNN to evaluate if the target is in the frame and perhaps how much of the target is in the frame. This allows an extra step of particle selection where particles without the target object or with less of the target object are eliminated before evaluating for the affine parameters. With this elimination, the values in the rules to reject less target-represented chosen image patches being introduced into the dictionary could be further refined.

An implementation of a pre-trained CNN could also improve the tracker's ability to handle illuminations and occlusions. As the number of layers in a CNN increases, CNN's abilitiy to differentiate of the target and the non-target objects improves, regardless of the illuminations and occlusion on the target object.

5.3 RGB Video Frames

As mentioned in section 2.1, gray-scale image retains much less information than a red-green-blue (RGB) image. Since IVT, LSST and RVT in their current state can only evaluate gray-scale videos, some of the information may have been lost. For instance, a bright yellow target with a light green background converted into gray-scale becomes similar shades of gray, and this could lead to target boundary box shifting or target misidentification. An few extra steps of calculations could be introduced to allow the evaluation of an RGB image to strengthen some of the target features, which may improve the tracking performance.

Bibliography

[1] C. Apostolopoulos, K. Nasrollahi, M. H. Yang, M. N. S. Jahromi, and T. B. Moeslund. Occlusion-aware pedestrian detection, Mar. 2019.

[2] P. B. Belhumeur and D. Kriegman. What is the set of images of an object under all possible lighting conditions? In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 1996.

[3] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, Jan. 1998.

[4] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. *ECCV 2016: Computer Vision, Part of the Lecture Notes in Computer Science book series (LNCS)*, 9905:749–765, Sep. 2016.

[5] T. Jebara. Generative versus discriminative learning. *Machine Learning. Part of the International Series in Engineering and Computer Science.*, 755:17–60, 2004.

[6] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale 1-regularized least squares. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):606–617, Dec. 2007.

[7] K. Koh, S.-J. Kim, and S. Boyd. l1_ls: Simple matlab solver for l1-regularized least squares problems. https://web.stanford.edu/ boyd/l1_ls/, Apr. 2008. Hosted by Stanford University.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012.

[9] B. K. S. Kumar, Swamy, M. N. S., and M. O. Ahmad. Weighted residual minimization in pca subspace for visual tracking. *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2016.

[10] M. Landy and J. A. Movshon. *The Plenoptic Function and the Elements of Early Vision*, chapter 1, pages 3–20. MIT Press, Cambridge, Massachusetts, 1991.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov. 1998.

[12] A. Levey and M. Lindenbaum. Sequential karhunen-loeve basis extraction and its application to images. *IEEE Transactions on Image Processing*, 9(8):1371–1374, Aug. 2000.

[13] X. Mei and H. Ling. Robust visual tracking and vehicle classification via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11), Nov. 2011.

[14] M. Merler, N. Ratha, R. S. Feris, and J. R. Smith. Diversity in faces. *Computing Research Repository (CoRR)*, abs/1901.10436, Jan. 2019.

[15] B. Moghaddam and A. Pentland. Probabilistic visual learning for object detection. In *Proceedings of IEEE International Conference on Computer Vision*, pages 786–793, Jun. 1995.

[16] H. Murase and S. K. Nayar. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, Jan. 1995.

[17] A. L. of Ordinary Videos. Amsterdam library of ordinary videos dataset. https://www.alov300.org. Hosted by Amsterdam Library of Ordinary Videos.

[18] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141, May 2007.

[19] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. http://www.cs.toronto.edu/ dross/ivt/, Aug. 2011. Hosted by University of Toronto.

[20] S. T. Roweis. Em algorithms for pca and spca. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 626–632. MIT Press, 1998.

[21] I. W. Selesnick. The estimation of laplace random vectors in additive white gaussian noise. *IEEE Transactions on Signal Processing*, 56(8):3482–3496, Aug. 2008.

[22] W. Shen, Y. Wu, J. Yuan, L. Duan, J. Zhang, and Y. Jia. Robust distracter-resistive tracker via learning a multi-component discriminative dictionary. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(7):2012–2028, Jul. 2019.

[23] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. H. Lau, and M.-H. Yang. Vital: Visual tracking via adversarial learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8990–8999, Jun. 2018.

[24] Y. Sui and L. Zhang. Robust tracking via locally structured representation. *International Journal of Computer Vision*, 119(2):110–144, Sep. 2016.

[25] C. Sun, F. Li, H. Lu, and G. Hua. Visual tracking via joint discriminative appearance learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2567–2577, Dec. 2017.

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, Jun. 2015.

[27] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[28] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 61(3):611–622, Oct. 1999.

[29] P. Tseng. Convergence of a block-coordinate descent method for a undifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, Jun. 2001.

[30] D. Wang, H. Lu, and M.-H. Yang. Robust visual tracking via least soft-threshold squares. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(9):1709–1721, Sep. 2016.

[31] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031–1044, Apr. 2010.

[32] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):210–227, Apr. 2008.

[33] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *Journal ACM Computing Surveys (CSUR)*, 38(4), Dec. 2006.

[34] X. Zhou, L. Kou, H. Ding, X. Fu, and Y. Shang. Similarity learning for template-based visual tracking. *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pages 1–6, Jul. 2014.

[35] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. *Computing Research Repository (CoRR)*, abs/1707.07012, Jul. 2017.

Appendix A

Proofs of Lemmas

Both proofs are as presented by Ross et al. [18].

A.1 Proof of Lemma 1

Lemma 1 addressed in section 2.1.1: The SKL Algorithm has the following proof:

By definition,

$\bar{\mathbf{I}}_C = \frac{n}{n+m}\bar{\mathbf{I}}_A + \frac{m}{n+m}\bar{\mathbf{I}}_B$

$\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_C = \frac{m}{n+m}\left(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B\right)\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_C = \frac{n}{n+m}\left(\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_A\right)$

And

$$
\begin{aligned}
\mathcal{S}_C &= \sum_{i=1}^{n} \left(\mathbf{I}_i - \bar{\mathbf{I}}_C\right)\left(\mathbf{I}_i - \bar{\mathbf{I}}_C\right)^\top + \sum_{i=n+1}^{n+m} \left(\mathbf{I}_i - \bar{\mathbf{I}}_C\right)\left(\mathbf{I}_i - \bar{\mathbf{I}}_C\right)^\top \\
&= \sum_{i=1}^{n} \left(\mathbf{I}_i - \bar{\mathbf{I}}_A + \bar{\mathbf{I}}_A - \bar{\mathbf{I}}_C\right)\left(\mathbf{I}_i - \bar{\mathbf{I}}_A + \bar{\mathbf{I}}_A - \bar{\mathbf{I}}_C\right)^\top + \sum_{i=m+1}^{n+m} \left(\mathbf{I}_i - \bar{\mathbf{I}}_B + \bar{\mathbf{I}}_B - \bar{\mathbf{I}}_C\right)\left(\mathbf{I}_i - \bar{\mathbf{I}}_B + \bar{\mathbf{I}}_B - \bar{\mathbf{I}}_C\right)^\top \\
&= \mathcal{S}_A + n\left(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_C\right)\left(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_C\right)^\top + \mathcal{S}_B + m\left(\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_C\right)\left(\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_C\right)^\top \\
&= \mathcal{S}_A + \frac{nm^2}{(n+m)^2}\left(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B\right)\left(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B\right)^\top + \mathcal{S}_B + \frac{n^2m}{(n+m)^2}\left(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B\right)\left(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B\right)^\top \\
&= \mathcal{S}_A + \mathcal{S}_B + \frac{nm}{n+m}\left(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B\right)\left(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B\right)^\top
\end{aligned}
$$

A.2 Proof of Lemma 2

Lemma 2 addressed in section 2.1.1: The Forgetting Factor has the following proof:

When a forgetting factor of f is used, the incremental PCA algorithm in Figure 2.4 computes the left-singular vectors $\mathbf{U}'$ and singular values $\mathbf{\Sigma}'$ of the matrix $[f\mathbf{U\Sigma}\ \ \hat{\mathbf{B}}]$. This is equivalent to computing the eigenvectors and (the square-roots of) the eigenvalues of $\left[f\mathbf{U\Sigma}\ \ \hat{\mathbf{B}}\right]\left[f\mathbf{U\Sigma}\ \ \hat{\mathbf{B}}\right]^\top$. Now,

$$
\begin{aligned}
\left[f\mathbf{U\Sigma}\ \ \hat{\mathbf{B}}\right]\left[f\mathbf{U\Sigma}\ \ \hat{\mathbf{B}}\right]^\top &= f^2\mathbf{U\Sigma^2U}^\top + \hat{\mathbf{B}}\hat{\mathbf{B}}^\top \\
&= f^2\mathbf{U\Sigma V}^\top\mathbf{V\Sigma}^\top\mathbf{U}^\top + \hat{\mathbf{B}}\hat{\mathbf{B}}^\top \\
&= f^2\left(\mathbf{A} - \bar{\mathbf{I}}_\mathbf{A}\right)\left(\mathbf{A} - \bar{\mathbf{I}}_\mathbf{A}\right) + \hat{\mathbf{B}}\hat{\mathbf{B}}^\top \\
&= f^2\mathcal{S}_A + \mathcal{S}_B + ct
\end{aligned}
$$

Where ct is a correction term that adjusts the mean of the eigenbasis, and $\mathcal{S}_A$ and $\mathcal{S}_B$ are scatter matrices — a scalar times the covariance matrix — as defined in Lemma 1.

Appendix B

Proof of Equation 2.5

This is a proof of $\mathcal{N}\left(\mathbf{I}_t; \mu, \mathbf{U}\mathbf{U}^\top + \varepsilon\mathbf{I}\right) \propto (\mathbf{I}_t - \mu)^\top \left(\mathbf{U}\mathbf{U}^\top + \varepsilon\mathbf{I}\right)^{-1} (\mathbf{I}_t - \mu)$ in Equation 2.5. Let $\mathbf{y}$ denote $(\mathbf{I}_t - \mu)$,

$$\begin{aligned}(\mathbf{I}_t - \mu)^\top \left(\mathbf{U}\mathbf{U}^\top + \varepsilon\mathbf{I}\right)^{-1} \mathbf{y} &= \mathbf{y}^\top \left[\varepsilon^{-1}\mathbf{I} - \varepsilon^{-1}\mathbf{U}\left(\mathbf{I} + \varepsilon^{-1}\mathbf{U}^\top\mathbf{U}\right)^{-1}\mathbf{U}^\top\varepsilon^{-1}\right]\mathbf{y} \\ &= \mathbf{y}^\top\varepsilon^{-1}\left(\mathbf{I} - \frac{\varepsilon^{-1}}{1+\varepsilon^{-1}} \cdot \mathbf{U}\mathbf{U}^\top\right)\mathbf{y} \\ &= \varepsilon^{-1}\mathbf{y}^\top\left(\mathbf{I} - \frac{\varepsilon^{-1}}{1+\varepsilon^{-1}} \cdot \mathbf{U}\mathbf{U}^\top\right)\mathbf{y},\end{aligned} \tag{B.1}$$

where $\varepsilon^{-1}\mathbf{y}^\top\left(\mathbf{I} - \frac{\varepsilon^{-1}}{1+\varepsilon^{-1}} \cdot \mathbf{U}\mathbf{U}^\top\right)\mathbf{y}$ is the exact probabilistic principle component analaysis (PPCA). It is easy to recognize the proportionality of

$$\varepsilon^{-1}\mathbf{y}^\top\left(\mathbf{I} - \frac{\varepsilon^{-1}}{1+\varepsilon^{-1}} \cdot \mathbf{U}\mathbf{U}^\top\right)\mathbf{y} \propto \mathbf{y}^\top\left(\mathbf{I} - \frac{1}{1+\varepsilon} \cdot \mathbf{U}\mathbf{U}^\top\right)\mathbf{y}. \tag{B.2}$$

Furthermore, in incremental PCA in IVT with the assumption that ε is approximately zero,

$$\begin{aligned}\mathbf{y}^\top\left(\mathbf{I} - \frac{1}{1+\varepsilon} \cdot \mathbf{U}\mathbf{U}^\top\right)\mathbf{y} &\overset{\varepsilon\longrightarrow 0}{\cong} \mathbf{y}^\top\left(\mathbf{I} - \mathbf{U}\mathbf{U}^\top\right)\mathbf{y} \\ &= \mathbf{y}^\top\left(\mathbf{I} - \mathbf{U}\mathbf{U}^\top\right)\left(\mathbf{I} - \mathbf{U}\mathbf{U}^\top\right)\mathbf{y} \\ &= \left\|\mathbf{y}^\top - \mathbf{U}\mathbf{U}^\top\mathbf{y}\right\|^2.\end{aligned} \tag{B.3}$$

Appendix C

MATLAB Code

C.1 Fitness of Evaluation of Particles During Tracking

```
function [x, s, L, it] = vt_lasso(y,A,ls,lx,sgnx,x,s)
% function [x, s, L, it] = vt_lasso(y,A,ls,lx,sgnx,x,s)
%
% It solves min_{x,s} L=(||A*x-y-s||_2)^2+lx*||x||_1+ls*||s||_1
%
% NOTES: y can be a matrix in which case the problem is solved
%        for each of its columns
%        ls > 0; if ls=Inf--> s=0
%        lx, sgnx are optional
%        inf > lx >= 0 or lx=[] --> lx=0
%        sgnx = 1--> x>=0 or sgnx=-1,0,[]--> x is real-valued
% SPECIAL CASES (examples)
% 'IVT': lx=0; ls=inf; sgnx=[];
% 'RVT': lx=ls=0.1; sgnx=1;
% 'LSST': lx=0; ls=0.1; sgnx=[];
%
% % Created by Athanasios Sideris, Aug. 2019

if nargin < 4
  lx=0.1;
  sgnx=[];
elseif nargin < 5
  sgnx=[];
end
if isempty(lx)
  lx=0;
end
sgnx=sign(sgnx);

T=A'*A;
```

```
a=diag(T);
P=T\A';

[m,n]=size(A);
k=size(y,2); % number of cases
if nargin < 6
  % initialize with the LS solution
  Q=[T+lx*eye(n) -A';-A (ls+1)*eye(m)];
  z=Q\[A'*y;y];
  x=z(1:n,:);
  s=z(n+1:end,:);
elseif nargin < 7
  disp('Both x0 and s0 are required')
  return
end

bx=zeros(n,k);
bs=zeros(m,k);
L  = vecnorm(y,2).^2;
iy = true(1,k); %y-cases that have not yet converged
it = 1;
while any(iy) && it < 2000
  L0=L;
  % fix s---update x
  if lx==0,
    x(:,iy)=P*(y(:,iy)+s(:,iy));
  else % iterate over x-coordinates
    bx0(:,iy)=-2*A'*(y(:,iy)+s(:,iy));
    for i=1:n
      i1=setdiff((1:n),i);
      bx(i,iy)=T(i,i1)*x(i1,iy)+bx0(i,iy);
      if sgnx > 0
        x(i,iy)=max(bsxfun(@rdivide,-(bx(i,iy)+lx), 2*a(i)),0);
      else
        x(i,iy)=-max(bsxfun(@rdivide,abs(bx(i,iy))-lx,2*a(i))
           ,0).*sign(bx(i,iy));
      end
    end
  end
  % fix x---update s
  if isinf(ls)
    s=zeros(m,k);
  else
    bs(:,iy)=(A*x(:,iy)-y(:,iy));
    s(:,iy)=max(abs(bs(:,iy))-ls/2,0).*sign(bs(:,iy));
  end
  % compute objective function
  L(:,iy)=vecnorm(A*x(:,iy)-y(:,iy)-s(:,iy),2).^2+lx*vecnorm(x
     (:,iy),1);
  if ~isinf(ls)
    L(:,iy)=L(:,iy)+ls*vecnorm(s(:,iy),1);
  end
  dLa = L-L0;
  dLr = dLa./L0;
  iy = max(abs([dLa;dLr])) >= 1e-5; %y-cases not yet converged
  it = it + 1;
end
if it==1000
  disp('Max number of iterations exceeded in vt_lasso')
end
```

C.2 Proposed Modifications

The following MATLAB routines are tracking modifications proposed to the routines proposed by Ross et al. [19]. The authorship is noted in each routine to indicate whether it is an adaptation and modification or it is created for the purpose of the tracking modification proposal.

C.2.1 Main Routine

Prior to running the tracking routine, these requirements must be fulfilled:

- All frames in the video to be tracked must be transformed into gray-scale, each pixel with a value ranging from 0 to 1 (0 being black and 1 being white).
- The video has to be stored under the variable "data" with the rows as pixels along x-coordinate, the columns entries as pixels along y-coordinate, and the third dimension being the frames of the video.
- The location of the initial rectangular target boundary box has to be chosen and stored as "p" with entries p_x, p_y, s_x, s_y, and θ, each denoting the target boundary box center x-coordinate in pixels, center y-coordinate in pixels, length in x-axis in pixels, length in y-axis in pixels, and angle rotation in radians.
- Optional: The ground truth points in each frame is stored as "truepts" and are defined with rows as x-coordinate of each point, columns as y-coordinate of each point, and the third dimension being the frames of the video.

```
1  % DESCRIPTION OF OPTIONS:
2  %
3  % Following is a description of the options you can adjust for
4  % tracking, each proceeded by its default value.  For a new
       sequence
5  % you will certainly have to change p.  To set the other
       options,
6  % first try using the values given for one of the demonstration
7  % sequences, and change parameters as necessary.
8  %
9  % p = [px, py, sx, sy, theta]; The location of the target in
       the first
10 % frame.
11 % px and py are th coordinates of the centre of the box
```

```
12 % sx and sy are the size of the box in the x (width) and y (
       height)
13 %   dimensions, before rotation
14 % theta is the rotation angle of the box
15 %
16 % 'numsample',400,   The number of samples used in the
       condensation
17 % algorithm/particle filter.  Increasing this will likely
       improve the
18 % results, but make the tracker slower.
19 %
20 % 'condenssig',0.01,  The standard deviation of the observation
        likelihood.
21 %
22 % 'ff',1, The forgetting factor, as described in the paper.
       When
23 % doing the incremental update, 1 means remember all past data,
        and 0
24 % means remeber none of it.
25 %
26 % 'batchsize',5, How often to update the eigenbasis.  We've
       used this
27 % value (update every 5th frame) fairly consistently, so it
       most
28 % likely won't need to be changed.  A smaller batchsize means
       more
29 % frequent updates, making it quicker to model changes in
       appearance,
30 % but also a little more prone to drift, and require more
       computation.
31 %
32 % 'affsig',[4,4,.02,.02,.005,.001]  These are the standard
       deviations of
33 % the dynamics distribution, that is how much we expect the
       target
34 % object might move from one frame to the next.  The meaning of
        each
35 % number is as follows:
36 %
37 %    affsig(1) = x translation (pixels, mean is 0)
38 %    affsig(2) = y translation (pixels, mean is 0)
39 %    affsig(3) = rotation angle (radians, mean is 0)
40 %    affsig(4) = x scaling (pixels, mean is 1)
41 %    affsig(5) = y scaling (pixels, mean is 1)
42 %    affsig(6) = scaling angle (radians, mean is 0)
43 %
44 % CORRECTED
45 %     affsig(1) = x translation (pixels, mean is 0)
46 %     affsig(2) = y translation (pixels, mean is 0)
47 %     affsig(3) = overall scaling by ratio (both x and y)
48 %     affsig(4) = rotation in radians (~75 samples reaches 1
       sigma, 7500 samples reaches 2 sigma)
49 %     affsig(5) = elongation scaling by ratio,
50 %     affsig(6) = skew-ness, scaling angle (uncertain units,
       but [0 1], mean is 0)
51 %
52 % OTHER OPTIONS THAT COULD BE SET HERE:
53 %
54 % 'tmplsize', [32,32] The resolution at which the tracking
```

```
    window is
% sampled, in this case 32 pixels by 32 pixels.  If your
    initial
% window (given by p) is very large you may need to increase
    this.
%
% 'maxbasis', 16 The number of basis vectors to keep in the
    learned
% apperance model.
%
% script: runtracker.m
% requires:
%   data(h,w,nf)
%   param0
%   opt.tmplsize [h,w]
%      .numsample
%      .affsig [6,1]
%      .condenssig
%
% % Copyright (C) Jongwoo Lim and David Ross.
% % All rights reserved.
% % SPECIFY PARAMETERS
% % Modified by Athanasios Sideris and Yu Hua N. Tseng

% opt INITIALIZATION
rand('state',0);  randn('state',0);
if ~exist('opt','var')          opt = []; end
if ~isfield(opt,'tmplsize')     opt.tmplsize = [32,32]; end %
    resolution of the image patches
if ~isfield(opt,'numsample')    opt.numsample = 800; end % number
    of particles
if strcmp(datatitle,'01-Light_video00001')
  opt.lassolam = 0.1;
  opt.affsig = [9,9,0.01,0.01,0.002,0.001]; % variances of
     affine parameters
elseif strcmp(datatitle,'davidin300')
  if ~isfield(opt,'affsig')     opt.affsig =
     [3,3,.01,.02,.002,.001]; end
elseif strcmp(datatitle,'car4') || strcmp(datatitle,'dudek')
  if ~isfield(opt,'affsig')     opt.affsig =
     [3,3,.08,.01,.002,.001]; end
  opt.lassolam = 0.1;
elseif strcmp(datatitle,'dudek')
  if ~isfield(opt,'affsig')     opt.affsig =
     [3,3,.08,.01,.002,.001]; end
end
if ~isfield(opt,'condenssig') opt.condenssig = 0.01; end
if ~isfield(opt,'maxbasis')   opt.maxbasis = 16; end % number
    of eigenbasis templates
if ~isfield(opt,'batchsize')  opt.batchsize = 5; end % number
    of frames to be remembered before updating eigenbasis
if ~isfield(opt,'errfunc')    opt.errfunc = 'L2'; end
if ~isfield(opt,'ff')         opt.ff = 1; end % 1 = forget
    nothing
if ~isfield(opt,'gamma')      opt.gamma = 0.5;  end % for error
    function in condensation
if ~isfield(opt,'lassolam')   opt.lassolam = 0.5;  end % for
```

```
    LASSO
if ~isfield(opt,'maxbasis2')  opt.maxbasis2 = 8;  end % number
    of image patch templates

% TRANSFORMING param0
param0 = [p(1), p(2), p(3)/32, p(5), p(4)/p(3), 0]; clear('p')
param0 = affparam2mat(param0); % initial target affine

frame = double(data(:,:,1))/256; % initial frame, [0,1]

% tmpl INITIALIZATION
tmpl.mean = warpimg(frame,param0,opt.tmplsize); % target image
    patch
tmpl.basis = []; % initialize
tmpl.eigval = []; % initialize
tmpl.numsample = 0; % initialize
tmpl.reseig = 0; % initialize residual eigenvalues
sz = size(tmpl.mean);
N = sz(1)*sz(2); % total number of pixels in each video frame

% param INITIALIZATION
param = []; % initialize param structure
param.est = param0; % initial target region parameters
param.wimg = tmpl.mean; clear('param0') % initial basis mean

% truepts TRACKING
if (exist('truepts','var'))
  npts = size(truepts,2);
  aff0 = affparaminv(param.est);
  pts0 = aff0([3,4,1;5,6,2])*[truepts(:,:,1);ones(1,npts)];
  pts = cat(3,pts0+repmat(sz'/2,[1,npts]),truepts(:,:,1));
  trackpts = zeros(size(truepts));
  trackerr = zeros(1,npts); meanerr = zeros(1,npts);
else
  pts = [];
end; clear('aff0')

%% RUN TRACKING AND UPDATING
% draw initial track window
drawopt = drawtrackresult([],0,frame,tmpl,param,pts);
disp('resize the window as necessary, then press any key..');
    pause;
drawopt.showcondens = 0;  drawopt.thcondens = 1/opt.numsample;

wimgs = [];

% initialized values are from frame 0
duration = 0; tic;
if (exist('dispstr','var')) dispstr=''; end

param.ref = param.wimg; param.conf_rec = []; tmpl.dict2 = [];
for f = 1:size(data,3)
  frame = double(data(:,:,f))/256;

  % do tracking
  param = estwarp_condens_prop(frame,tmpl,param,opt);

  % do update
```

```
152   if ~isempty(param.recon)
153     if size(tmpl.dict2,2) >= opt.maxbasis2
154       tmpl.dict2(:,1) = [];
155     end
156     tmpl.dict2 = [tmpl.dict2,param.wimg(:)];
157   end
158   wimgs = [wimgs, param.recon(:)];
159   if (size(wimgs,2) >= opt.batchsize) && ~isempty(wimgs)
160     if (isfield(param,'coef'))
161       ncoef = size(param.coef,2);
162       recon = repmat(tmpl.mean(:),[1,ncoef])+tmpl.basis*param.
            coef;
163       [tmpl.basis,tmpl.eigval,tmpl.mean,tmpl.numsample] = ...
164         sklm_prop(wimgs,tmpl.basis,tmpl.eigval,tmpl.mean,tmpl.
              numsample,opt.ff);
165       param.coef = tmpl.basis'*(recon-repmat(tmpl.mean(:),[1,
            ncoef]));
166     else
167       [tmpl.basis,tmpl.eigval,tmpl.mean,tmpl.numsample] = ...
168         sklm_prop(wimgs,tmpl.basis,tmpl.eigval,tmpl.mean,tmpl.
              numsample,opt.ff);
169     end
170     wimgs = [];
171
172     if size(tmpl.basis,2) > opt.maxbasis
173       tmpl.reseig = opt.ff^2*tmpl.reseig+sum(tmpl.eigval(opt.
            maxbasis+1:end).^2);
174       tmpl.basis = tmpl.basis(:,1:opt.maxbasis);
175       tmpl.eigval = tmpl.eigval(1:opt.maxbasis);
176       if (isfield(param,'coef'))
177         param.coef = param.coef(1:opt.maxbasis,:);
178       end
179     end
180   end
181
182   duration = duration + toc;
183   % draw result
184   if (exist('truepts','var'))
185     trackpts(:,:,f) = param.est([3,4,1;5,6,2])*[pts0;ones(1,
            npts)];
186     pts = cat(3,pts0+repmat(sz'/2,[1,npts]),truepts(:,:,f),
            trackpts(:,:,f));
187     idx = find(pts(1,:,2)>0);
188     if (length(idx)>0)
189       trackerr(f) = sqrt(mean(sum((pts(:,idx,2)-pts(:,idx,3))
              .^2,1)));
190     else
191       trackerr(f) = nan;
192     end
193     meanerr(f) = mean(trackerr(~isnan(trackerr)&(trackerr>0)));
194     if (exist('dispstr','var')) fprintf(repmat('\b',[1,length(
            dispstr)])); end;
195     dispstr = sprintf('%d: %.4f / %.4f',f,trackerr(f),meanerr(f
            ));
196     fprintf(dispstr);
197   end
198   tic;
199
```

```
200   % figure(1) current frame, truepts, trackpts, patches ----
201   fig1 = figure(1);clf;
202   axes('Position', [0 1/2 1 1/2])
203   imshow(double(data(:,:,f))/256); % draw current frame
204   hold on;
205   drawbox(opt.tmplsize, param.est, 'Color','r', 'LineWidth'
          ,2.5) % draw patch box
206   plot(trackpts(1,:,f),trackpts(2,:,f),'r+','MarkerSize',6) %
          draw truepts
207   plot(truepts(1,:,f),truepts(2,:,f),'y+','MarkerSize',6) %
          draw truepts
208   text(0,0,num2str(f),'FontSize',15,'Color','y','FontWeight','
          bold','VerticalAlignment','top')
209
210   axes('Position', [0 1/3 1 1/6])
211   imshow([tmpl.mean,param.wimg]) % current image patch
212   text(2,0,'basis mean','FontSize',8,'Color','y','
          VerticalAlignment','top')
213   text(opt.tmplsize(1),0,'current patch','FontSize',8,'Color','
          y','VerticalAlignment','top')
214
215   axes('Position', [0 1/18 1 1/3])
216   I1 = []; I2 = []; I3 = []; I4 = [];
217   for basis = 1:size(tmpl.basis,2)
218     if basis <= 8
219       I1 = [I1 (reshape(tmpl.basis(:,basis),opt.tmplsize)-min(
              tmpl.basis(:,basis)))/(max(tmpl.basis(:,basis))-min(
              tmpl.basis(:,basis)))];
220     elseif basis >= 9 && basis <= 16
221       I2 = [I2 (reshape(tmpl.basis(:,basis),opt.tmplsize)-min(
              tmpl.basis(:,basis)))/(max(tmpl.basis(:,basis))-min(
              tmpl.basis(:,basis)))];
222     end
223   end
224   if size(I1,2)/opt.tmplsize(1) < 8 % gray out unfilled
          templates
225     I1 = [I1 0.5*ones(opt.tmplsize(1),(opt.tmplsize(2)*8 - size
          (I1,2)))];
226   end
227   if size(I2,2)/opt.tmplsize(1) < 8 % gray out unfilled
          templates
228     I2 = [I2 0.5*ones(opt.tmplsize(1),(opt.tmplsize(2)*8 - size
          (I2,2)))];
229   end
230   I = [I1; I2; I3; I4]; clear('I1','I2','I3','I4','basis')
231   imshow(I)
232   for basis = 1:size(tmpl.basis,2) % template number text
233     if basis <=8
234       text((basis - 1)*opt.tmplsize(1),0,num2str(basis),'Color'
              ,'y','FontSize',10,'VerticalAlignment','top')
235     elseif basis > 8 && basis <= 16
236       text((basis - 9)*opt.tmplsize(1),opt.tmplsize(2),num2str(
              basis),'Color','y','FontSize',10,'VerticalAlignment','
              top')
237     end
238   end
239
240   axes('Position', [0 0 1 1/9]) % image patch dictionary
241   I = [];
```

```
242     for basis = 1:size(tmpl.dict2,2)
243       I = [I,reshape(tmpl.dict2(:,basis),opt.tmplsize)];
244     end
245     if size(I,2)/opt.tmplsize(1) < opt.maxbasis2 % gray out
          unfilled templates
246       I = [I 0.5*ones(opt.tmplsize(1),(opt.tmplsize(2)*opt.
            maxbasis2 - size(I,2)))];
247     end
248     imshow(I); clear('I')
249     text(2,0,'image patch dictionary','Color','y','FontSize',10,'
          VerticalAlignment','top')
250
251     % saving figure -------------------------------
252     % record_dir = [''];
253     % if ~exist(record_dir) mkdir(record_dir); end
254     % saveas(fig1,[record_dir 'frame_' num2str(f) '.jpg'])
255   end
256   duration = duration + toc;
257   fprintf('\n%d frames took %.3f seconds: %.3f fps\n',f,duration,
        f/duration);
```

C.2.2 Particle Filtering

```
function param = estwarp_condens(frm, tmpl, param, opt)
% function param = estwarp_condens(frm, tmpl, param, opt)
%
% frm           current frame of the video
% param.param   particle affine parameters (mat)
% param.est     previous frame's target affine parameter (mat)
% param.conf    normal PDF likelihood (of particles)
% param.coef
% param.wimg    previous frame's target image patch (in resolution)
% param.recon   reconstructed LSST image patch (mean replacing error pixels)
% param.err
% tmpl.mean     basis mean (resolution size)
% tmpl.basis    eigenbasis templates in eigenbasis template dictionary
% tmpl.eigval   eigenvalues associated with eigenbasis templates
% opt.numsample number of particles
% opt.affsig    affine parameters
% opt.errfunc   choice likelihood PDF
% opt.rsig      variance for using robust PDF
% opt.condenssig  variance for condensation
%
% % Copyright (C) Jongwoo Lim and David Ross.
% % All rights reserved.
% % Modified by Athanasios Sideris and Yu Hua N. Tseng

n = opt.numsample; % number of particles
sz = size(tmpl.mean); % size of mean image (resolution)
N = sz(1)*sz(2); % total number of pixels in resolution

if ~isfield(param,'param')
  param.param = repmat(affparam2geom(param.est(:)), [1,n]);
else
  cumconf = cumsum(param.conf);
  idx = floor(sum(repmat(rand(1,n),[n,1]) > repmat(cumconf,[1,n ]))))+1;
  param.param = param.param(:,idx);
end
param.param = param.param + randn(6,n).*repmat(opt.affsig(:),[1,n]); % perturbation of the parameters (particles)
wimgs = warpimg(frm, affparam2mat(param.param), sz); % extract image patch particles
diff = repmat(tmpl.mean(:),[1,n]) - reshape(wimgs,[N,n]); % difference between particle abd basis mean

% PARTICLE WEIGHTS BASED ON AREA CHANGE --------
param_mat_prev = [param.est(1) param.est(2) param.est(3) 0 param.est(5) param.est(6)];
c_prev = corners(opt.tmplsize, param_mat_prev(:) , 'Color','r', 'LineWidth',2.5);
param_geom = affparam2geom(param_mat_prev(:));
A_prev = patch_area(param_geom(6),c_prev); c=[];
```

```
for count = 1:n
  c(:,:,count) = corners(opt.tmplsize, affparam2mat([param.
     param(1:4,count); 0; param.param(6,count)]), 'Color','r',
     'LineWidth',2.5);
  A(count) = patch_area(param.param(6,n),c(:,:,count));
  particle_weight(count) = 1/(1 + abs(1-A(count)./A_prev));
end; clear('c_prev','param_geom','A_prev','c','A') % ----

coefdiff = 0;
if (size(tmpl.basis,2) > 0) % active after the 1st batch
  y = diff;
  U = tmpl.basis;
  lx=0; ls=opt.lassolam; sgnx=[]; % ls = 0.1 for effective calc
  [coef,s,dist,it] = vt_lasso(y,U,ls,lx,sgnx);
  param.coef = coef;
end

if (~isfield(opt,'errfunc')) opt.errfunc = []; end
switch (opt.errfunc)
  case 'robust'
    param.conf = exp(-sum(diff.^2./(diff.^2+opt.rsig.^2))./opt.
       condenssig)';
  case 'ppca'
    param.conf = exp(-(sum(diff.^2) + sum(coefdiff.^2))./opt.
       condenssig)';
  otherwise
    if (size(tmpl.basis,2) > 0) % starting after 1st batch
      param.conf = exp(-opt.gamma*dist)';
    else
      param.conf = exp(-sum(diff.^2)./opt.condenssig)'; %
         likelihood in Gaussian PDF
      param.conf = param.conf.*particle_weight(:); % weighting
         likelihood
    end
end

[maxprob,maxidx] = max(param.conf); % find the highest
   likelihood
if (size(tmpl.basis,2) == 0)
  dist = zeros(1,n);
  s = zeros(1,n);
end
param.conf_rec = [param.conf_rec, [param.conf(maxidx); dist(
   maxidx);norm(s,1)]];

% USE IMAGE PATCH TEMPLATE DICTIONARY ---------
if (size(tmpl.basis,2) > 0) && dist(find(exp(-dist)==max(exp(-
   dist)))) > 2.8
  diff = repmat(tmpl.dict2(:,end),[1,n]) - reshape(wimgs,[N,n])
     ;
  y = diff;
  U = tmpl.dict2;
  lx=0; ls=opt.lassolam; sgnx=[]; % ls = 0.1 to have effective
     calc
  [coef,s,dist,it] = vt_lasso(y,U,ls,lx,sgnx);
  param.conf = exp(-opt.gamma*dist)';
  [maxprob,maxidx] = max(param.conf); % find the highest
     likelihood
end % ----------------------------------------

```

```
93  % REJECTING BAD IMAGES AS TEMPLATES ---------
94  bad_template = 0;
95  if dist(maxidx) > 2.8 %param.conf(maxidx) < 0.96
96  bad_template = 1;
97  end % ---------------------------------------
98
99  param.conf = param.conf ./ sum(param.conf); % normalize the
       probability to [0,1]
100 param.est = affparam2mat(param.param(:,maxidx)); % affine
       parameter of the chosen image patch
101 param.wimg = wimgs(:,:,maxidx); % the distorted particle with
       the highest likelihood
102 param.err = reshape(diff(:,maxidx), sz); % error of in the
       chosen image patch
103 if (size(tmpl.basis,2) > 0) % starting after 1st batch
104   param.recon = param.wimg;
105   if (sum(coef(:,maxidx)) ~= 0)
106     noisy_pixel = find(s(:,maxidx)~=0);
107     param.recon(noisy_pixel) = tmpl.mean(noisy_pixel);
108   end
109 else % before 1st batch
110   param.recon = param.wimg + param.err; % reconstructed image
111 end
112
113 % REJECTING BAD IMAGES AS TEMPLATES -----
114 if (bad_template == 1)
115   param.recon = [];
116 end % ----------------------------------
```

C.2.3 Target Boundary Box Area Calculation

```
function A = patch_area(theta,c)
%
% INPUT:
% theta  the skew angle [rad] (6th entry of affparam2geom)
% c      the corners of the target boundary box, 2-by-6, [x;y]
%
% OUTPUT:
% A      area of the image patch
%
% % Created by Yu Hua N. Tseng (Auguest 2019)

c(1,:) = c(1,:) - min(c(1,:)); % translation of x-axis
c(2,:) = c(2,:) - min(c(2,:)); % translation of y-axis
if (theta >= 0 && theta < pi/2) || (theta >= pi && theta < 3/4*pi) || (theta <= -pi/2 && theta > -pi) || (theta <= -3/4*pi && theta > -2*pi) % for positive phi
  theta_b = atan2(c(2,2),c(1,2)); % angle of base to translated x-axis
  theta_a = atan2(c(2,4),c(1,4)); % angle of slanted side to translated x-axis
  theta_a_rot = theta_a - theta_b; % angle of slanted side to translated and rotated x-axis
  base = sqrt(c(1,2)^2+c(2,2)^2);
  a = sqrt(c(1,4)^2+c(2,4)^2); % length of slanted side
elseif (theta >= pi/2 && theta < pi) || (theta >= 3/4*pi && theta < 2*pi) || (theta <= 0 && theta > -pi/2) || (theta <= -pi && theta > -3/4*pi) % for negative phi
  theta_b = atan2(c(2,1),c(1,2)-c(1,1)); % angle of base to translated x-axis
  theta_a = atan2(c(2,4)-c(2,1),c(1,1)); % angle of slanted side to translated x-axis
  theta_a_rot = theta_a - theta_b; % angle of slanted side to translated and rotated x-axis
  base = sqrt((c(1,2) - c(1,1))^2+c(2,1)^2);
  a = sqrt(c(1,1)^2+(c(2,4)-c(2,1))^2); % length of slanted side
end
height = a*sin(theta_a_rot);
A = abs(base*height);
```

C.2.4 Target Boundary Box Corners

```
function drawing_corners = corners(varargin)
% corners(opt.tmplsize, param_mat, 'Color','r', 'LineWidth
   ',2.5)
% function drawbox(width,height, param, properties)
%                  ([width,height], param, properties)
%
%   param, properties are optional
%
% INPUT:
% ([w,h] patch_resolution, param_in_mat, str, str, str, num)
%
% OUTPUT:
% drawing_corners    in format [corners_x;corners_y], 2-by-5
%
% % Copyright (C) Jongwoo Lim and David Ross.
% % All rights reserved.
% % Modified by Yu Hua N. Tseng

%--------------------------------------------------------------
% process the inputs
%--------------------------------------------------------------
if (length(varargin{1}) == 2) % opt.tmplsize
  w = varargin{1}(1);
  h = varargin{1}(2);
  varargin(1) = [];
else
  [w,h] = deal(varargin{1:2});
  varargin(1:2) = [];
end

if (length(varargin) < 1 || any(length(varargin{1}) ~= 6))
  M = [0,1,0; 0,0,1];
else
  p = varargin{1}; % opt.tmplsize
  if (length(varargin) > 1 && strcmp(varargin{2},'geom'))
    p = affparam2mat(p);
    varargin(1:2) = [];
  else
    varargin(1) = []; % opt.tmplsize
  end
  M = [p(1) p(3) p(4); p(2) p(5) p(6)];
end

%--------------------------------------------------------------
% box corners
%--------------------------------------------------------------
drawing_corners = [ 1,-w/2,-h/2; 1,w/2,-h/2; 1,w/2,h/2; 1,-w/2,
   h/2; 1,-w/2,-h/2 ]';
drawing_corners = M * drawing_corners;
```

Appendix D

Demonstration Example

To illustrate the processes in which IVT, LSST and RVT select the most likely estimated image patch, the dictionary templates and a set of 800 particles are drawn at a certain moment of the video are used for calculating the likelihood of each particles.

D.1 Existing Templates

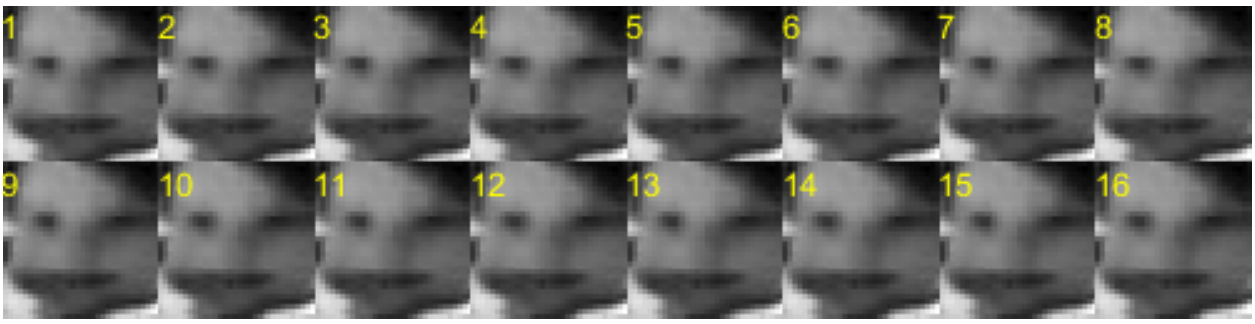


Figure D.1: Templates available in the dictionary for the example calculations.

D.2 Drawn Particles

Figure D.2: The example illustration of the target boundary boxes of the particles drawn (yellow boxes) and the estimated target boundary box of the previous frame (red box).

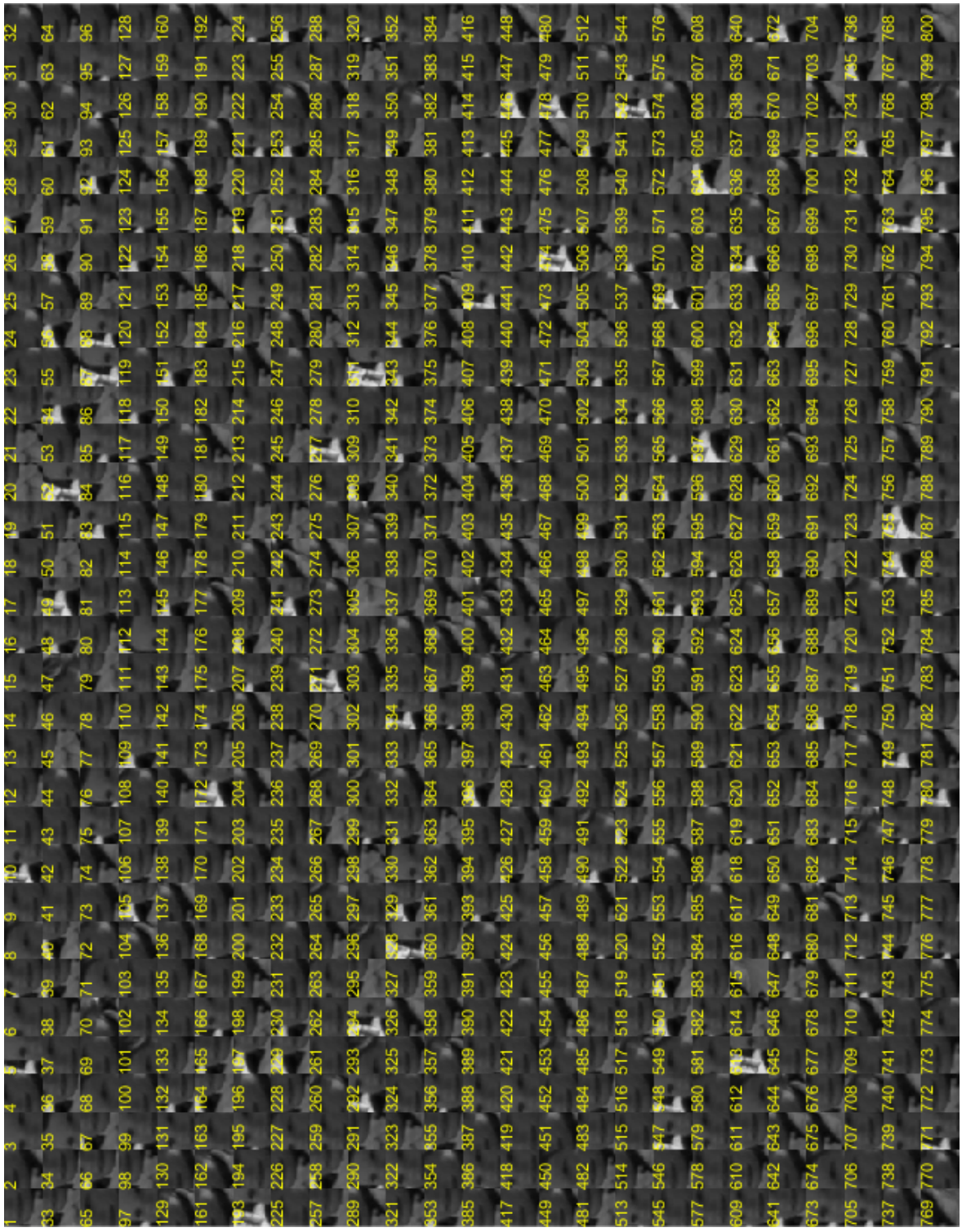


Figure D.3: All the image patch particles drawn for the example calculations.

Appendix E

Approach Comparison and Summary

		IVT	LSST	RVT
Paper Setting (common)	**Number of dictionary templates**, n_t	16	16	10
	Image patch resolution (w-by-h)	32×32	32×32	12×15, 15×12, 10×18, or 18×10
	Number of particles, n_p	600	600	300
	Affine parameter variances $(x, y, \theta, s, \alpha, \phi)$	(set by case)	(set by case)	(0.06, 0.01, 0.01, 0.06, 4, 4)

Table E.1: Essential settings to define for IVT, LSST and RVT that are common in all tracking approaches.

	IVT	LSST	RVT
Settings (deviated)	Batch size (dictionary update): 5	Batch size: 5	σ in Equation 2.35: 10^{-4}
	Probability function: normal PDF	λ in Equation 2.23: 0.1	Number of velocities to average: 3
	Forgetting factor: 0.95	γ in Equation 2.25: 0.5	ℓ_1 solver: Koh et al.[7]
	Cut-off eigenvalue for templates: 10^{-6}		λ in Equation 2.30 unspecified
			Similarity cut-off τ in step 6 of Figure 2.9: 0.5

Table E.2: Essential settings to define for IVT, LSST and RVT that are specific to each tracking approach.

<table>
<tr><th>IVT</th><th>LSST</th><th>RVT</th></tr>
<tr><td colspan="3">1. Initialize with the first target image patch in the first frame as time $t = 0$, acquiring:
· Affine parameters of frame 0, $\mathbf{x}_t$
· Target image patch in specified resolution of frame 0 $\mathbf{y}_t \in \mathbb{R}^{d\times 1}$ (d is number of pixels in the resolution size)</td></tr>
<tr><td colspan="2">2. –</td><td>2. Populate all dictionary templates in the specified resolution and drawn from $\mathbf{y}_t$ with the variance σ in Equation 2.35, acquiring:
· Dictionary templates $\mathbf{U} = [\mathbf{a}_1 \cdots \mathbf{a}_{n_t}] \in \mathbb{R}^{d\times n_t}$
· Template weights, $\mathbf{w} = \{\|\mathbf{a}_i\|_2\}_{i=1,\cdots,n_t}$</td></tr>
<tr><td colspan="3">3. Draw particles, acquiring:
· Affine parameters of the image patch particles $\mathcal{X}_t = \left[\mathbf{x}_t^{(1)} \cdots \mathbf{x}_t^{(n_p)}\right]$
· Image patch particles $\mathcal{Y}_t = \left[\mathbf{y}_t^{(1)} \cdot \mathbf{y}_t^{(n_p)}\right] \in \mathbb{R}^{d\times n_p}$</td></tr>
<tr><td>4. Find the distances d_w and d_s $\in \mathbb{R}^{n_t\times 1}$ for all particles Equation 2.7, acquiring
· $d_s^{(i)} = \mathbf{U}^\top \left(\mathbf{y}^{(i)} - \boldsymbol{\mu}\right), i = 1, \cdots, n_p$
· $d_w^{(i)} = \left(\mathbf{y}^{(i)} - \boldsymbol{\mu}\right) - d_s^{(i)}$
5. Choose the particle with the highest likelihood $p\left(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}\right)$ by Equation 2.3 and $\hat{\mathbf{y}} = \max_{i=1,\cdots,n_p} p\left(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}\right)$</td><td>4. Formulate the LSS optimization in Equation 2.24 and solve for $\hat{\mathbf{z}}$ and $\hat{\mathbf{s}}$ and distance $d_{LSS}(\mathbf{y}; \mathbf{U})$ of Equation 2.19, $i = 1, \cdots, n_p$, acquiring
· Coefficients $\hat{\mathbf{z}}^{(i)} \in \mathbb{R}^{d\times 1}$
· Errors $\hat{\mathbf{s}}^{(i)} \in \mathbb{R}^{d\times 1}$
· Distance $d_{LSS}^{(i)} \in \mathbb{R}^1$
5. Choose particle with the highest likelihood $p\left(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}\right) = e^{\gamma d^{(i)}}$ and $\hat{\mathbf{y}} = \max_{i=1,\cdots,n_p} p\left(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}\right)$</td><td>4. Formulate the optimization of Equation 2.30 to solve for the coefficients $\mathbf{x}^{(i)}$ and trivial coefficients $\{\mathbf{r}^-\}^{(i)}$ and $\{\mathbf{r}^-\}^{(i)}$, $i = 1, \cdots, n_p$, acquiring
· $\mathbf{x}^{(i)}$
· $\{\mathbf{r}^+\}^{(i)}$
· $\{\mathbf{r}^-\}^{(i)}$
5. Choose the particle with the highest likelihood $p\left(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}\right)$ of Equation 2.35 and $\hat{\mathbf{y}} = \max_{i=1,\cdots,n_p} p\left(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}\right)$</td></tr>
<tr><td colspan="2">6. Update $\mathbf{U}$ and $\boldsymbol{\mu}$ with incremental PCA algorithm in Figure 2.3</td><td>6. Update dictionary $\mathbf{U}$, template weights $\mathbf{w}$ and normalized templates $\mathbf{a}_{i=1,\cdot,n_t}$ by dictionary template update algorithm from step 3 in Figure 2.9</td></tr>
<tr><td colspan="3">7. Repeat from step 2.</td></tr>
</table>

Table E.3: Key steps in IVT, LSST and RVT.

Appendix F

Target Boundary Box Area Calculation

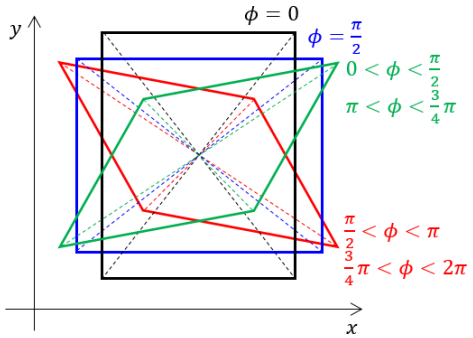
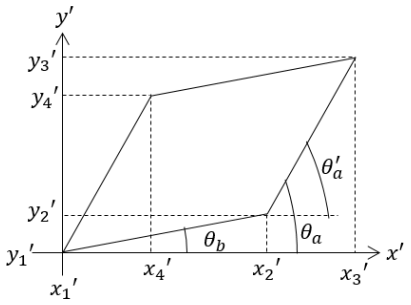
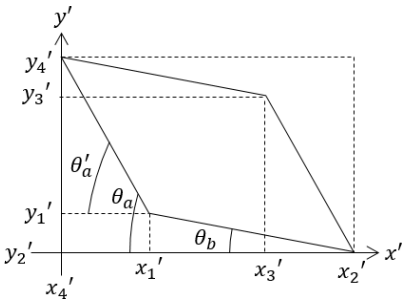


Figure F.1: A illustration of how a target boundary box gets skewed if starting from the black rectangle with $\phi = 0$ [rad]. The red boundary box is a result when ϕ is $(0, \frac{\pi}{2})$ or $(\pi, \frac{3\pi}{4})$, the blue boundary box the result when $\phi = \frac{\pi}{2}$, and the green boundary box is a result when ϕ is $(\frac{\pi}{2}, \pi)$ or $(\frac{3\pi}{4}, 2\pi)$. All skewed boundary boxes all share a common center and the same size of enclosed area.



(a) Image patch with skew angle ϕ when ϕ is $(0, \pi)$ [rad].



(b) Image patch with skew angle ϕ when ϕ is $(\pi, 2\pi)$ [rad].

Figure F.2: Examples of a skewed rectangular target boundary box translated to the new axes x' and y'.

The area inside the target boundary box bounded by (x_1, y_1), (x_2, y_2), (x_3, y_3) and (x_4, y_4) is computed. The information known about the boundary box is the coordinates of the four corners. The calculation of the area is only illustrated for the white boundary box in Figure F.2a, since the case in Figure F.2b has the same idea.

The coordinates of the corners are translated to the new coordinates:

$$\begin{aligned}(x_1', y_1') &= (x_1 - x_1, y_1 - y_1)\\ (x_2', y_2') &= (x_2 - x_1, y_2 - y_1)\\ (x_3', y_3') &= (x_3 - x_1, y_3 - y_1)\\ (x_4', y_4') &= (x_4 - x_1, y_4 - y_1)\end{aligned} \tag{F.1}$$

The information needed to compute for the area of the target boundary box, which is a parallelogram, is the base (the distance between (x_1', y_1') and (x_2', y_2')) and the height (the distance between (x_4', y_4') and the base).

The angle between (x_2', y_2') and the x'-axis is denoted by θ_b, where

$$\theta_b = \tan^{-1} \frac{y_2'}{x_2'}. \tag{F.2}$$

This is the angle that rotates the whole parallelogram onto rotated axes.

The angle between the line formed by (x'_2, y'_2) and (x'_3, y'_3) and the x'-axis is denoted by θ_a, where

$$\theta_a = \tan^{-1} \frac{y'_4}{x'_4}. \tag{F.3}$$

The angle between the line formed by (x'_2, y'_2) and (x'_3, y'_3) and the rotated rotated x'-axis is required, which is denoted by θ'_a. It can be found by

$$\theta'_a = \theta_a - \theta_b. \tag{F.4}$$

The length of the base b is between (x'_1, y'_1) and (x'_2, y'_2), and can be found by

$$b = \sqrt{(x'_2 - x'_1)^2 + (y'_2 - y'_1)^2}. \tag{F.5}$$

The length a between (x'_1, y'_1) and (x'_4, y'_4) is needed to find the height of the parallelogram, which is

$$a = \sqrt{(x'_4 - x'_1)^2 + (y'_4 - y'_1)^2}. \tag{F.6}$$

Hence, the height h can be found by

$$h = a \cdot \sin\theta'_a. \tag{F.7}$$

And the area of the parallelogram $A_{parallelogram}$ is

$$A_{parallelogram} = b \cdot h. \tag{F.8}$$