Tracking with Context

by

Carlo du Toit

Thesis presented in fulfilment of the requirements for the degree of

Master of Science in the Faculty of Applied Mathematics

at the Stellenbosch University

Department of Mathematical Sciences (Division Applied Mathematics),
Stellenbosch University,
Private Bag X1, Matieland 7602, South Africa.

Supervisors:

Dr M.R. Hoffmann Prof. B.M. Herbst

December 2016

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: December 2016 .....

Abstract

Tracking with Context

C. du Toit

Department of Mathematical Sciences (Division Applied Mathematics),
Stellenbosch University,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MSc (Applied Mathematics)

December 2016

Tracking in an unconstrained environment presents a difficult challenge. Abrupt object motion, appearance changes, non-rigid objects and occlusion are but a few of the trials faced. To overcome these challenges a tracking algorithm, often an unsupervised learning problem, should be fast and capable of on-the-fly modelling. A huge variability in the range of input observation data is to be expected. Generative tracking models are good with dealing with unsupervised learning, but not always trustworthy without good verification, which leads to drifting.

In this thesis we investigate the effectiveness of integrating knowledge (context) into the tracking model and to provide verification to generative models, improving the drifting problem and trustworthiness of the model. To

accomplish this we implement a model based on what we learn from other context aware implementations.

Our model is context flexible, capable of integrating any existing object detector, providing the model with valuable knowledge. Experimentation shows it is capable of integrating with any target tracker, and provides valuable assistance in the form of verification. When the target undergoes aggressive appearance changes, gets fully occluded or even leave the field of view, our model is capable of tracking the target successfully until the main tracker can resume its task. Context is not only there to serve the main target tracker, but also to improve learning of the model itself. We use the model to minimise the possibility of a miss-match during training itself, providing increased certainty.

Uittreksel

Beeldsporing met Visuele Konteks

("Tracking with Context ")

C. du Toit

Departement van Wiskundige Wetenskappe (Afdeling Toegepaste Wiskunde),
Stellenbosch Universiteit,
Privaatsak X1, Matieland 7602, Suid Afrika.

Tesis: MSc (Toegepaste Wiskunde)

Desember 2016

Die spoor van 'n voorwerp in 'n onbeperkte omgewing het baie uitdagings. Skielike beweging, voorkoms verandering, nie-rigiede voorwerpe en okklusies is slegs 'n paar van die moontlike uitdagings. Om hierdie uitdagings te oorkom moet 'n sporingsalgoritme vinnig wees en oor die vermoë beskik om intyds te kan modelleer. Dikwels word dit as 'n leerprobleem sonder toesig hanteer. Groot variasie in die invoer waarnemingsdata is te verwagte. Generatiewe modelle is goed om te leer sonder toesig, maar nie altyd betroubaar sonder goeie verifikasie nie. Dit lei tot dryf.

In hierdie tesis ondersoek ons die effektiwiteit om kennis (konteks) in die sporingsmodel te integreer. Verder bied dit verifikasie vir generatiewe mo-

delle. Verifikasie verminder die kans vir dryf en verbeter die betroubaarheid van die model. Ons implementeer 'n model gebaseer op vroeëre kontekssporingsalgoritmes.

Ons model is onafhanklik van die spesifieke konteks en in staat om met enige bestaande voorwerpherkenner te kan integreer, sodoende die model te verskaf met waardevolle kennis/konteks. Eksperimentele resultate toon aan dat ons model integreer met enige bestaande sporingsimplementasie en in staat is om waardevolle verifikasie vir sporingsalgoritmes te bied. Wanneer die teikenvoorwerp van voorkoms verander, volledige okklusie ondergaan of selfs die beeldraam verlaat, is ons model in staat om die teiken te volg totdat die sporingsimplementasie weer oorneem. Konteks is nie slegs daar om die sporingsimplementasie te help nie, maar ook om die leerproses van die model te verbeter. Ons gebruik die model om die moontlikheid van 'n herkenningsfout tydens die leerfase te verminder, en so dan die akkuraatheid te verhoog.

Erkenning

Ek wil graag my dankbaarheid en waardering aan McElory gee vir sy insig en leiding. Ek het McElory aan die begin van my studies genader met twee doelwitte, om dit deeltyds te doen en vir my liefde vir Toegepaste Wiskunde. Ek kan met vertroue beaam dat my doelwitte bereik is. My waardering gaan ook uit aan Prof Herbst en die departement van Toegepaste Wiskunde.

Laastens en nie die minste, aan my vrou, Rika, en my dogter, Izandri, vir hul liefde en ondersteuning deur al die chaos met die geboorte van ons eersteling, die tyd vir my gegee het om my liefde vir wiskunde te kon uitleef.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Visual tracking

Tracking is a fundamental problem of computer vision with various applications in the modern world. Major advances in mathematics and hardware specification in the computing industry give room for a lot of growth in the field of tracking. Typical challenges in the field of visual tracking vary from abrupt object motion, appearance changes, changes in scene illumination, camera motion, tracking non-rigid objects and in particular tracking objects through occlusion.

Merging the concepts from the surveys in both Yilmaz *et al.* [2] and Yang *et al.* [3], the typical process flow for a visual tracking method is described in Figure 1.1.

As per Yilmaz *et al.* [2] understanding all the complexities in tracking an

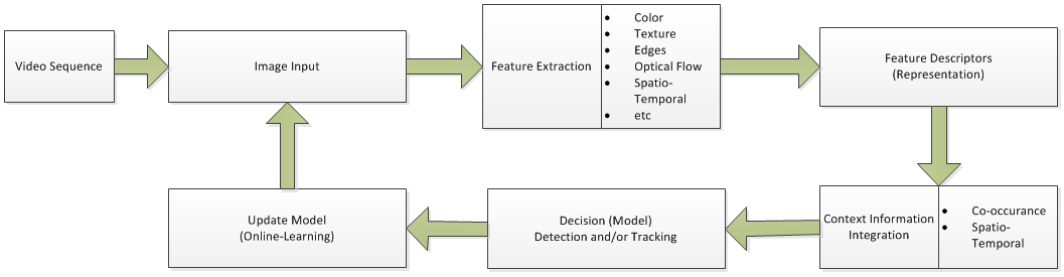


Figure 1.1: Common flow diagram in tracking algorithms

object(s), the methods differ in the way they answer the following three questions:

- which object representation to use (for example points, contour, probability densities);
- which image features to use (for example color, edges, texture, gradient) and
- how should motion, appearance, and shape of the object be modelled.

The choice of object representation and which features to use are often closely related, for example choosing a specific representation like contours forces one to sometimes use specific features like edges. In the survey by Jalal and Singh [4], features and object representation are all part of the building block called object modelling. In Li *et al.* [5], another survey, appearance modelling consists of the two building blocks, object representation and statistical modelling.

Every tracking algorithm has some form of object detection/tracking mechanism in every frame or in the first appearance of the object. The object detection model needs to represent deformation of the object (appearance and shape) while the object tracker needs to model the motion. The choice of object model ultimately plays a key role in the detection/tracking algorithm to use, especially when it comes to how we represent the appearance and shape of the object.

Given a sequence of images, temporal information exists and some object detectors are capable of using this information to update their model. Temporal information of an object also gives valuable motion information and this gets modelled by the object tracker. The task of the object detector and object tracker can be performed separately or jointly and both need to build on the choice we make for our object model.

Most statistical models to accomplish the task of object detection and/or object tracking, can be categorised as either generative (descriptive), discrimi-

native or hybrid generative-discriminative. To get a more detailed explanation of these types of models, a recommended read is Bishop and Lasserre [6] and Bishop [7]. A quick summary follows in Section 1.2.

It is clear that no matter what each survey defines the different components in the architecture of visual tracking as, they are all closely dependent on each other. All these building blocks need to be fully understood before researching the field of visual tracking.

Given the dynamic, ever changing nature when it comes to tracking in an unconstrained environment, we need modelling methods that are adaptive, capable of learning and updating their representations online. The object model has a key task of ensuring the object is distinguished from others. The challenge is finding the right balance between robustness and accuracy. Improving accuracy requires incorporating more specific techniques and features; this however reduces the generalisation of a model. Improving robustness requires for more relaxed modelling constraints, resulting in less accurate results. The one comes at a price of the other.

Unlike the surveys discussed up to now, in Smeulders *et al.* [8] they assess trackers against a more vast set of sequences. To really understand the robustness of a tracker, they present a variety of challenges in their test data. They focused on short but many sequences. In conclusion it was found that the top performing trackers had no common underlying method. There is still no best single solution for a generalised tracker, capable of tracking in an unconstrained environment. When a tracker focus on one particular constraint, performance on other constraints decreases.

1.2 Modelling

In machine learning the goal often is to take an input vector (observation) x_i and then assign it to a specific class label c_i out of a set of labels $C = [c_1, ..., c_n]$, giving us the typical classification problem. Statistically the classification, regression or prediction problem, is about calculating the posterior conditional distribution $p(C|x_i)$. Given a trained model, one can predict

the class $c_i \in C$ given the value of an observation vector x_i of input features. Typical models are generative, discriminative or hybrid generative-discriminative with each contributing both advantages and disadvantages.

1.2.1 Discriminative models

These models describes the system as a probabilistic or a non-probabilistic model. For example a support vector machine, classified as a discriminative model, is a non-probabilistic binary linear classifier. When the model is defined as a probabilistic model, it typically assumes the task of classification that directly models the posterior conditional distribution, $p(C|x_i)$. It aims to maximise the separability between the object and non-object regions discriminatively, a binary classification approach. No underlying probability distributions are modelled to generate the posterior conditional distribution, and for this reason it is sometimes considered a black-box model. Relationships between variables is not explicit as in the generative model. Discriminative methods require proper training data to optimally separate the classes and this is a time-consuming exercise.

Examples of discriminative models are, logistic regression, neural networks and conditional random fields.

1.2.2 Generative (Descriptive) models

Generative models describe the system as a probabilistic model, modelling the distribution of both the input and output as a joint probability $p(C, x_i)$, where in the typical classification problem, x_i is the input or observation variable, and C the output or set of class labels. To turn this into a classifier we use Bayes' rule,

$$p(C|x_i) = \frac{p(x_i|C)p(C)}{p(x_i)}.$$

On the right hand side we need the prior $p(C)$ and the class-conditionals $p(x_i|C)$, i.e. a model for each class. The distribution $p(x)$ is simply a normalisation.

Estimating the class-conditional densities for each class makes this model more computationally intense; the more unique classes the more computations. One often models unnecessary characteristics and structures in the class-conditional densities that adds little value to the posterior conditional distribution, $p(C|x_i)$. This is wasteful if the end goal is basic classification.

Generative models facilitates the generation of synthetic data by sampling from the class-conditionals as well as making predictions, given underlying distributions are modelled. The fact that we can insert knowledge on the relationship between variables, dependencies and prior knowledge, is what makes the joint distribution so adaptable. Examples of generative models are the Naïve Bayes Classifier, Gaussian Mixture Models (GMM), Hidden Markov Models (HMM). In Table 1.1, we summarise some of the advantages and disadvantages of generative models.

Advantages	**Disadvantages**
Ability to introduce prior knowledge	Potential wasteful modelling
Do not require large datasets	Reliant on domain expertise
Generation of synthetic inputs	Do not scale to large number of classes
Easy to train	Vulnerable to drift

Table 1.1: Generative models - advantages and disadvantages

To summarise, generative models train fast with small amount of data, but prediction is more computational. Discriminative models trains slowly, but prediction is fast.

Given the objectives of this thesis set out in Section 1.5 on Page 7, the requirements of tracking in an unconstrained environment and the fact that it is often an unsupervised learning problem, fast on-the-fly modelling is a necessity. A huge variability in the range of input/observations makes it difficult to almost impossible to provide enough labelled data to train properly. In the event of unlabelled data, as is the case with fast on-the-fly modelling, generative methods seem the direction to take for our specific objectives. It is however important to note with generative models, having online-update mechanisms, they tend to drift.

1.3 Problem statement

As mentioned at the start of this chapter, abrupt object motion, appearance changes, non-rigid objects and occlusion are just a few, but the most significant challenges we face in unconstrained environments. Over the years the field of visual tracking has produced a number of methods to track in these environments. Adding to the list of challenges, fast on-the-fly modelling and if possible no offline training, increases the difficulty of any modelling algorithm further. Fast on-the-fly modelling inherently pushes one to generative models. Generative models however, due to their online-updating capabilities are vulnerable to drift, giving a false positive (whether object being tracked is in fact the target object). Generative models are untrustworthy without good verification.

The problem we are trying to solve in this thesis is to track in an unconstrained environment using generative models and to avoid drifting by improving the way we verify our model estimate. A key future direction that was noted in the three surveys by Yilmaz *et al.* [2],Yang *et al.* [3] and Jalal and Singh [4], is the integration of knowledge (visual context) available in each frame. These pieces of context hold valuable information of great use in the task of tracking in unconstrained environments and will help solve the trustworthiness of generative models by means of good verification. See the context integration phase in the visual tracking flow diagram in Figure 1.1.

1.4 Literature synopsis

We dedicate Chapter 2 as a background study into some methods using context, in some form or another in the tracking problem as well as the task of object detection.

In Yang *et al.* [3] and Li *et al.* [5], it is worth noting that another future direction is the need to bring both the generative and discriminative models together, called hybrid generative-discriminative models, and use the best of both worlds. This however is outside the scope of this thesis and

the main focus in the background study is the integration of knowledge (context). It has already been noted that a hybrid generative-discriminative method does not necessarily guarantee better results than those of the individual models. It sometimes introduces constraints and more parameters to model, limiting its practicality for a flexible model.

1.5 Objectives

Given the problem statement and future directions specified in Section 1.4, we focus this thesis on building/improving a generalised model capable of tracking in an unconstrained environment, using generative model techniques for on-the-fly learning, that capitalises on the benefits of integrating knowledge (context). Here are the key objectives of this thesis:

- tracking in unconstrained environments,
- using models capable of on-the-fly learning from unlabelled data,
- improve trustworthiness of generative models and to avoid drifting,
- improve on the methods integrating knowledge (context) by means of higher-level scene understanding and
- define a model/framework that easily integrates and enhance existing tracking methods.

We want to better understand the existing methods for integrating context and in particular implementing and improving on the work done by Grabner *et al.* [9], using concepts and ideas from some of the other methods. Given the large number of references to the work done by Grabner *et al.* [9] in this thesis, we will refer to it as Grabner. The model we chose to improve on was selected for its simplicity, elegance and potential for easy integration of higher-level scene understanding.

1.6 Contributions

Here are a few of the contributions made in this thesis in addition to the work done by Grabner *et al.* [9]:

- Context flexible model that can deal with different types of context, prioritising the integration of real-world predictable and reliable context by means of weighted queues. This provides the option to use any off-the-shelf object detector, increasing higher-level scene understanding. Any number of object detectors can be used. The model falls back on low-level generic context in the event of failure to find any real-world objects. See Sections 3.2 on Page 39, 3.4.2 on Page 42, 3.4.5 on Page 48 and 4.3 on Page 68.
- Integrated the implementation of Alexe *et al.* [10], a class-less object detector as our default real-world supporter model. Improved the reliability of returned boxes containing objects in Alexe *et al.* [10] using K-means to cluster the best groups of potential areas. See Sections 3.4.8 on Page 52 and 4.3 on Page 68.
- Introduced relative awareness between the context and the target. Context can also benefit from the model during the detection phase and not just the target. It minimises the potential to miss-match context in subsequent frames causing noise in the final voting. See Sections 3.4.2.1 on Page 44 and 4.6 on Page 82.
- Management of stale supporters and removing them from the model allows for less computation and also the potential for outdated context to contribute to the estimate, should they suddenly appear. We use the term supporters to describe context that is useful in predicting the target object positions. See Sections 3.4.2.2 on Page 45 and 4.7 on Page 83.
- Improved the trustworthiness of the target tracker and to avoid drifting by means of verification. The model does not only have to be used during occlusion or when the main target tracker fails. It can also be

used to verify whether the main target tracker can be trusted. See Sections 3.4.3 on Page 45 and 4.2.2 on Page 56.

- Increased learning by doing so on every iteration. There is much to learn about interplay between context and the target even when both are standing still. See Section 3.4.4 on Page 46.
- Focused on quality context rather than quantity by means of a Gaussian priority filter. Context with the greatest correlation to the target gets a higher weighting in the voting mechanism. Also having two separate Gaussian filters, one for first-level and a second with reduced amplification for second-level supporters allowed for first-level supporters to be trusted for as long as they are present before giving over to second-level supporters. See Sections 3.4.6 on Page 49 and 4.5 on Page 81.

1.7 Outline of this thesis

In Chapter 1 on Page 1, we briefly covered and summarised the field of visual tracking and particularly the need for tracking in an unconstrained environment. A brief insight into the challenges, different statistical methods and future directions, helped define specific objectives we want to accomplish in this thesis. In short we need to improve the trustworthiness of generative models capable of learning on-the-fly by means of verification. Integrating knowledge into the model provides the necessary verification and reduces the chance of generative models drifting. Knowledge also compliments the task of tracking in unconstrained environments.

Chapter 2 on Page 11 provides a deeper understanding of the benefits of introducing context into the tracking problem, looking at some existing methods. In the summary Section 2.3 on Page 33, the consensus is clear that the benefits are invaluable, but higher-level scene understanding is required to further improve the stability and reliability of models integrating context.

In Chapter 3 on Page 36, we propose our improved model built on the work of Grabner by adding room for higher-level scene understanding and a few

additional improvements.

Chapter 4 on Page 55 dedicates to test results showing the benefits of the basic supporter model described in Chapter 3, and showing further results on the improvements we made as part of this thesis.

In Chapter 5 on Page 87, final conclusions and future directions are provided.

Chapter 2

Background

Numerous successes have been made integrating visual context in both object detection as well as visual tracking algorithms. The principle behind the idea of using visual context in tracking is the many temporary, but yet potential strong links that exist between the object being tracked and the visual context within the image. To get a better appreciation for the use of context, it is also worth understanding its uses in object detection. A good overview of the latter is Divvala *et al.* [11].

To explain visual context in a real-world practical example, consider tracking a person in a crowded area surrounded by many other individuals. A human will immediately recognise a few distinct properties like colour of the individuals clothes, a hat, gender, length, even notice another person walking next to the target individual, like a child holding his hands. All these things serve as visual cues and play an integral role when trying to keep track, keeping your eyes on the target individual as he moves through the crowd. The target might turn direction changing his appearance, or the target can walk behind a larger and taller individual, resulting in partial or full occlusion.

In these scenarios it is in our nature as humans to make use of the visual cues mentioned above, to try and keep an eye on the tracked person or at a minimum making a guess where he might be. In the event where the target faces another direction, items such as clothes, a hat, and so forth, will

be good visual cues since they have a correlated motion trajectory with the target. In the event of occlusion, visual cues like the child from our earlier example can be used to predict the potential target position until the target reappears. Using visual cues (context) is exactly the natural/practical way of thinking that an end-to-end tracking algorithms should benefit from.

In Yang *et al.* [3], the integration of context into a visual tracking method gets categorised under the following three approaches:

- co-occurrence,
- spatio-temporal relation, and
- knowledge.

Taking a closer look into a few existing implementations in this chapter, it either results in modelling the relative position and motion of the context to the target, or information that can help verify or improve the estimation confidence. In this chapter we categorise the implemented models as either positional or informational. We also discuss and compare a few existing methods that fall into one of these categories.

Looking at positional models, of particular note is the work done by Grabner and Yang *et al.* [12] which extended their own ideas from Yang *et al.* [13]. In Zhang *et al.* [14] they use adaptive correlation filters and Li and Nevatia [15] focus on a full spatio-temporal implementation.

Informational models using the nature of context to improve the estimation confidence, we discuss the work done by Hoiem *et al.* [1]. It is worth noting that Hoiem *et al.* [1] incorporates context into the task of object detection and not in tracking itself. In Section 1.1 it was pointed out that object detection plays an integral part of the end-to-end tracking algorithm and it is for that reason we focus on the entire process trying to see where possible we can capitalise on the use of context.

2.1 Positional modelling

In this category the positional properties of context get modelled as part of the object detector, tracker and/or verification phase. The essence lies in the spatial (intra-frame) information that exists between all objects present in the frame, and also the temporal (inter-frame) properties that exists in a sequence of frames, like motion correlation learned from frequent co-occurrence of objects from frame to frame.

2.1.1 Supporter model

Strong links exist between the object being tracked and the context in the image. Grabner exploits this principle of co-occurrence. They learned a model inspired by the Implicit Shape Model (ISM) algorithm, Leibe *et al.* [16], where local image features (context) vote for the object position. A brief summary of Implicit Shape Modelling is covered in Section 3.1.1 on Page 36.

Grabner track all local image features (SIFT descriptors) and calculate the relative distance and angle of these features to the target centre, updating the model on-the-fly. These image features are known as **supporters** and are defined as follow: *'Supporters are features which are useful to predicting the target object position. They at least temporarily move in a way which is statistically related to the motion of the target.'*

The main contrast to the ISM algorithm is the continued update of image features in the model and the potential links between them, i.e., if a supporter's motion to the target becomes uncorrelated, that supporter is no longer used in the voting.

Experiments showed that having few but accurate information sources leads to better results. It is ultimately the highly effective voting mechanism that is loosely based on the Generalised Hough Transform (GHT) that manages the supporters by associating and disassociating them with the target.

Their implementation leaves room for the use of any tracker. When the

target gets partially/fully occluded, or even under appearance changes, the supporter model is capable of tracking the target with good results until visible again.

Although not implemented or realised, the model can be used as a verification mechanism to the main tracker when the target is visible. This is something we capitalised on in our model. See Section 3.4.3 on Page 45.

Looking at the mathematical model they propose, the main focus is to model the probability of the target position x in image I_t, at time stamp t. This gives

$$p(x|I_t) = \frac{p(x, I_t)}{p(I_t)}$$

and by marginalising supporters s into the model we get

$$p(x|I_t) = \frac{\sum_{s \in S} p(x, I_t, s_1, .., s_i)}{p(I_t)},$$

where $s = s_1, .., s_i \in S$ is the set of supporters. Using basic statistics, we get

$$p(x|I_t) \propto \sum_{s \in S} p(x|s)p(s|I_t).$$

Given that the core implementation of this thesis builds on the model from Grabner, we give more detail on how we simplify the model above and how it gets put to use in Chapter 3 on Page 36. For our purpose in this chapter, we only need to know that the model has two distinct terms. Firstly the indicator function represented by the distribution $p(s|I_t)$, indicates the presence of a supporter in the image. Secondly the voting function represented by the distribution $p(x|s)$, contributes to the voting space $p(x|I_t)$, predicting where the target is.

Implementation of the indicator function gets accomplished by taking each supporter found in the current frame and matching it against a database of supporters captured over time. Having SIFT features for supporters, matching in the database is done using a dot-product of the normalised SIFT descriptors, and if above a set threshold θ, it is consider a match. A single descriptor can match multiple entries in the supporter database, so

the match with the highest score is selected. In addition to ensure more accurate matching, a simple KLT tracker is also used to establish supporter matches from two successive frames.

Each supporter successfully matched in the database needs to be updated as part of learning the model $p(x|s)$ when the target is present. Learning consists of updating for each matched supporter the following key parameters:

- r_i - the radius between supporter and target,
- ϑ_i - the angle between supporter and target, and
- Σ_i - the covariance matrix for the two variables r_i and ϑ_i.

The exponential forgetting principle is used when updating the model parameters, i.e., $r_t^{(i)} = \alpha r_{t-1}^{(i)} + (1-\alpha) r_t^{(i)}$ where i is the i-th supporter in the model, $\alpha \in [0,1]$.

In the event that the target cannot be found, the model is applied to calculate the voting space $p(x|I_t)$, predicting where the target is. The voting of a single supporter s_i is done approximating a single Gaussian

$$p(x|s_i) \propto \frac{1}{\sqrt{2\pi|\Sigma_i|}} e^{\left(-0.5(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)\right)}$$

where μ_i is the mean of the parameters r_i and ϑ_i stored as polar coordinates and Σ_i the covariance matrix of these parameters.

Figure 2.1 from Grabner's article best illustrates how the parameters are used to perform voting. The yellow dot represents a single supporter $x^{(i)}$ and x^* the target.

Each vote is cast in a Generalised Hough Transform space giving a two-dimensional map of potential target positions illustrated in Figure 2.2. Maximising this voting space gives the target position.

Figure 2.3 illustrates how the use of supporters can help track a target, both partially and fully occluded.

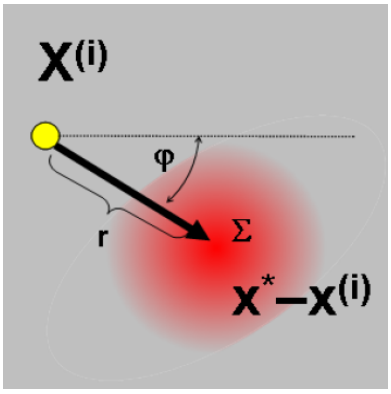


Figure 2.1: Voting parameters

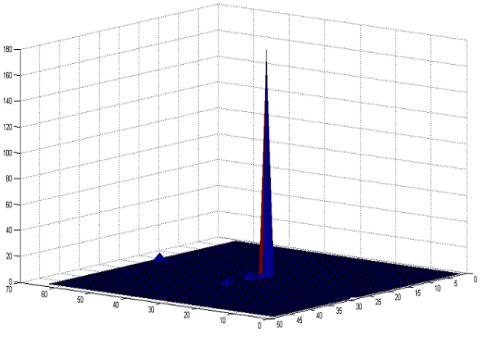
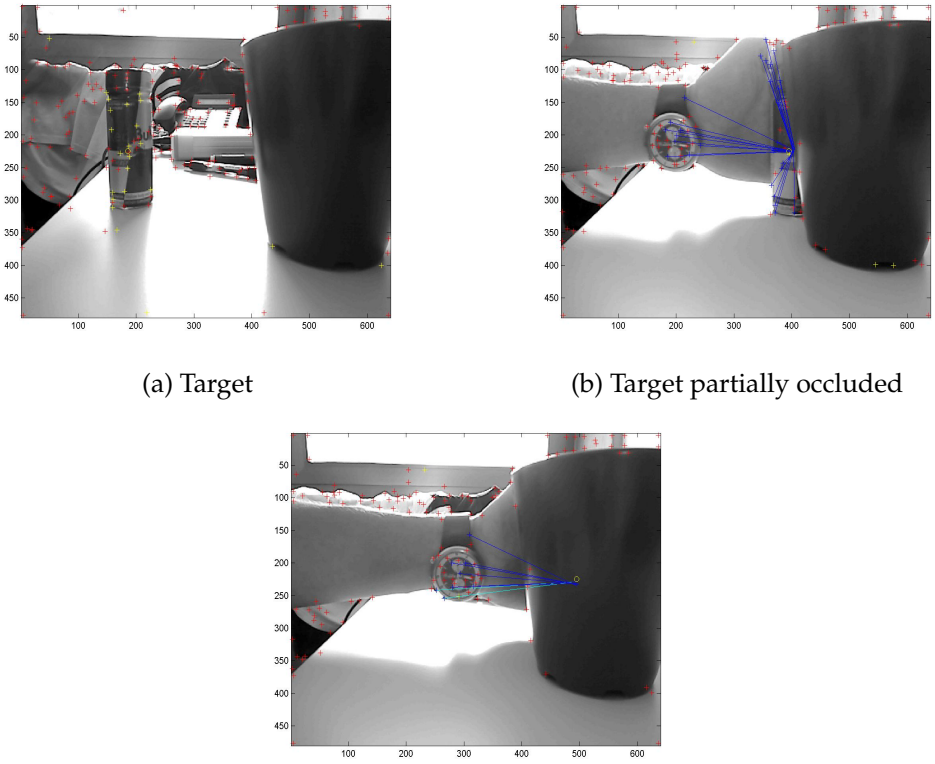


Figure 2.2: Generalised Hough Transform voting space

It is worth highlighting the following key properties that will be emphasised as we discuss other approaches throughout this background study. The context chosen in this approach is local feature points, giving the most localised information, but prone to occlusion and appearance changes. They do not capture object structure information and therefore not predictable in its motion. Co-occurrence is extremely important as there are so many supporters that it becomes computationally expensive to use context that provides no benefit. It is also essential that we only capture context that has predictable motion with the target.

A suggested read is the two articles Sun *et al.* [17] and Dinh *et al.* [18] that builds on the ideas in Grabner. In Dinh *et al.* [18] the supporter model gets optimised by only detecting and matching supporters around few candidates that has a high probability to be the target, instead of detecting and matching all potential supporters in the image. They also introduced the



(a) Target

(b) Target partially occluded

(c) Target fully occluded

Figure 2.3: Supporter model tracking under occlusion

concept of distractors. These are regions with similar appearance than the object being tracked causing drift. It is for that reason a tracker is employed for each of these distractor regions.

The implementation in Sun *et al.* [17], instead of choosing local feature points like SIFT, image regions are used. The context is called helpers instead of supporters and both the target and helpers can be tracked using existing tracking algorithms.

2.1.2 Region supporters

The essence of the Context-Aware Tracker in Yang *et al.* [12], also known as CAT throughout this chapter, is the combination of visual tracking and data mining, where data mining is used to discover auxiliary objects (context), by learning their co-occurrence associations and estimating affine motion

models to the target. It is the collaborative tracking of these auxiliary objects that leads to efficient and strong verification in the tracker.

The target and context is tracked as a random field, reducing the uncertainty of the tracker's estimation and effective verification by using the context as cues, due to strong motion correlation.

Auxiliary objects has to satisfy the following three properties:

- Frequent co-occurrence with target,
- Consistent motion correlation to the target, and
- Easy to track.

The data mining step learns a Markov random field (MRF) of spatial relationships between auxiliary objects and the target as a by-product, see Figure 2.4. The context in this implementation is once again low-level uncategorised features. Taking in consideration all the types of low-level features, we know feature points provide the best localised information, though as noted in Grabner, individual points are prone to occlusion and appearance changes. Feature points present computational challenges, and therefore instead the CAT implementation focuses on image regions. Image regions are less prone to occlusion and can be tracked using a mean-shift algorithm. The regions are pure colour segments and do not take texture (intensity variations of a surface) into consideration. This decision was made due to the computational constraint of bringing texture to the model.

The term auxiliary object should not be interpreted as actual objects as in the case of Li and Nevatia [15], where context is modelled at an object level, requiring additional level of abstraction. The objects in CAT are colour regions and not real-world objects. Procuring auxiliary objects start out with a set of candidate auxiliary objects. Candidate objects co-occur frequently with the target, but do not necessarily have strong motion correlation. A mean-shift tracker is used to track the objects (colour regions); if an object is lost for four consecutive frames, such an object is not considered a candidate

auxiliary object. Next step in the mining process is to find the true auxiliary objects having correlated motion with the target. This is accomplished by performing subspace analysis using the fact that when two points on two separate objects have affine motion relationship, they reside in a linear subspace. Identifying the subspace, estimates the affine motion model between an auxiliary object and the target.

To explain this briefly, imagine an auxiliary object x_{k_t} and target x_{v_t} at time t. We look at the trajectories for x_{k_t} over a time window $[t-M+1, t]$, where M represents the length of the sliding window. If we assume affine motion matrix A_t and translation vector b_t, then the following relationship exists

$$x_{v_t} = A_t x_{k_t} + b_t.$$

Subtract the mean $\bar{x}_{v_t}$ of x_{v_t} and $\bar{x}_{k_t}$ of x_{k_t}, over the window specified and taking zero mean white noise into consideration, the relationship can be expressed with $\widetilde{x}_{v_t} = x_{v_t} - \bar{x}_{v_t}$ and $\widetilde{x}_{v_t} = x_{k_t} - \bar{x}_{k_t}$ giving $\widetilde{x}_{v_t} = A_t \widetilde{x}_{k_t} + n$.

The key component lies in the subspace analysis of the covariance matrix $\widehat{C}$ of $\widetilde{x}_{v_t}$ and $\widetilde{x}_{k_t}$ defined as

$$\widehat{C} = \sum_{i=0}^{M-1} \begin{bmatrix} \widetilde{x}_{v_{t-i}} \\ \widetilde{x}_{k_{t-i}} \end{bmatrix} [\widetilde{x}^T_{v_{t-i}}, \widetilde{x}^T_{k_{t-i}}]$$

Performing eigenvalue decomposition on $\widehat{C}$ gives a sorted list of eigenvalues $\lambda_1, .., \lambda_4$. If there are more than two eigenvalues $\lambda_j^2 >> \sigma^2$ where σ represents the smallest eigenvalue, this candidate auxiliary object is not considered a true auxiliary object because the motion of the target and this object is not in one subspace. If it is considered an auxiliary object, using the property that the noise subspace is orthogonal to the signal subspace, the results from the eigenvalue decomposition gives means to estimate the variables A_t and b_t.

Now, considering all auxiliary objects have been identified at times step t, we can get rid of the time subscript and keep only the subscript k giving the index of a specific object. We now have the estimation parameters, A_k and

b_k for all auxiliary objects and using Gaussian distributions to characterise potentials between kth object and the target, we have a dynamic Markov random field (MRF) of auxiliary objects and the target. The pair-wise potentials are calculated as

$$\psi_{kv}(x_k, x_v) \propto e^{-\frac{(x_v - A_k x_k - b_k)^T (x_v - A_k x_k - b_k)}{2\sigma^2}}$$

where both A_k and b_k have been learned as a by-product of the mining process and represents the mean of the Gaussian for a specific potential k.

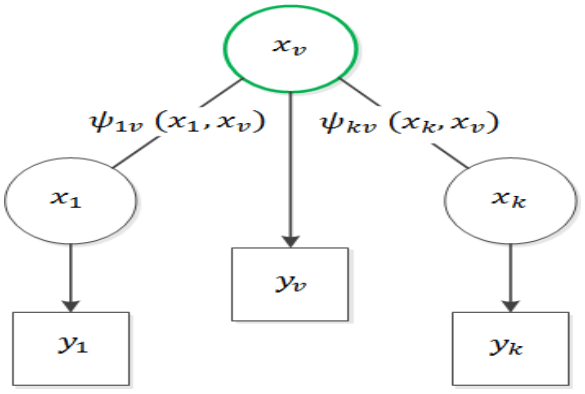


Figure 2.4: Markov random field - star topology

The structure of the MRF in Figure 2.4 is a star topology with pair-wise connectors only between target and object, no connection among auxiliary objects themselves. The random field is hidden and inferred from image evidence $\mathbf{Y}$. Given $\mathbf{Y} = y_k, k = 1, ..., K$ where y_v is the observation of x_v, the main purpose is to calculate the posteriors $p(x_v|\mathbf{Y})$ for the target and $p(x_k|\mathbf{Y})$ for all auxiliary objects. A belief propagation algorithm with two step message passing for non-loopy graphs gives an exact estimate of the posteriors. They use individual trackers for the target and auxiliary objects, but to benefit from collaborative tracking, they combine their individual estimates under the assumption that estimates are consistent. They refer to this combining of estimates as fusion.

The challenge with fusion is when the assumption of consistent estimation is incorrect and estimates from individual trackers are inconsistent. To make the fusion as robust as possible, they proposed a new theorem to measure

the consistencies for pair-wise Gaussian sources. Outliers are detected as a result of inconsistent estimates with all the others. One of three outcomes are possible based on the specific outlier.

- If the target is the outlier, drift or occlusion is most likely the cause and mining is suspended. Target estimation can be accomplished using estimates from the auxiliary objects.
- If an auxiliary object is the outlier, it is discarded from fusion and belief propagation is performed again, using the remaining Markov network.
- If the majority auxiliary objects are outliers, the target estimate is not verified and the tracker fails.

Both models in Yang *et al.* [12] and Grabner are complementary to the target tracker, and any choice of tracker can be used. Both models are trained when the target is visible and capable of predicting the target position when the main tracker fails. In Yang *et al.* [12] when the target is visible, the model is used as verification to the target tracker, reducing the uncertainty. The implementation in Grabner does not provide the same verification, but is capable of doing so with changes to the algorithm and training criteria.

In the CAT implementation when the target is assumed drifting or invisible, the mining process gets suspended; in Grabner a second-level supporter model gets initialised and learned, using context even when the target tracker fails. More information on second-level supporters are given in Section 3.4.7 on Page 51 where the implementation of the algorithm implemented by Grabner gets discussed in more depth.

The models of both Grabner and Yang *et al.* [12] are of a star topology with Gaussian potentials between context and the target. The main difference lies in the way they estimate. CAT uses belief propagation over a dynamic Markov random field and Grabner uses a voting technique in the Generalised Hough Transform space.

The context in CAT, like in Grabner, is still low-level and uncategorised in nature and not actual real-world objects. They do not fully capture object structure information and therefore less predictable in its motion. The use of image regions do however ensure for a more manageable number of context reducing the number of computations. These regions are also much less prone to occlusion increasing the chance of finding co-occurring features.

2.1.3 Integrating context with adaptive correlation filters

In Zhang *et al.* [14] we have another example of exploiting spatio-temporal context, using a Bayesian framework to model statistical correlation between low-level features on the target and local surrounding regions. This particular implementation was more focused on efficiency, unlike the other methods that are more computationally heavy and complex regarding feature extraction, detection and matching. This is accomplished by focusing on context close to the target and utilising efficient calculations in the frequency domain.

The core implementation, Bolme *et al.* [19] introduced a new type of filter, Minimum Output Sum of Squared Error (MOSSE), essentially an algorithm for producing stable correlation filters initialised from a single frame for the purpose of tracking. In Zhang *et al.* [14] they expand the MOSSE algorithm by extending the frame around the target to include valuable context.

Tracking the target is accomplished by maximising the confidence map

$$c(x) = p(x|o),$$

where x represents the object location, and o the presence of the object in the frame. The context feature set is defined as

$$X^c = c(z) = (I(z), z)|z \in \Omega_c(x^*),$$

where $I(z)$ denotes the image intensity at location z and $\Omega_c(x^*)$ the neighbourhood pixels of location x^*. Using the same Bayesian framework and principals in Grabner, they marginalise supporters (context) $c(z)$ into the model giving

$$\begin{aligned} p(x|o) &= \sum_{c(z)\in X^c} p(x, c(z)|o) \\ &= \sum_{c(z)\in X^c} p(x|c(z), o)p(c(z)|o), \end{aligned} \tag{2.1.1}$$

where $p(c(z)|o)$ is the context prior modelling the appearance of local context, and $p(x|c(z), o)$ the voting function, modelling the spatial relationship between the target and context as a probability.

In Bolme *et al.* [19], the voting function is nothing more than an adaptive filter that gets learned and updated from frame to frame. The voting function $p(x|c(z), o)$ is defined by means of the function

$$h^{sc}(x - z),$$

using relative distance and direction between the target x and the local context z. The function h^{sc} is described in Equation 2.1.4, and we first need to define more of the terms of the model below to better understand.

To satisfy the definition of probability, the context prior model $p(c(z)|o)$, much like Grabner, restricts the outcome to a range of zero to one, essentially stating the presence of context in the frame or not. To accomplish this they define

$$p(c(z)|o) = I(z)w_\sigma(z - x^*),$$

where $I(\cdot)$ is the image intensity at position z and $w_\sigma(\cdot)$ the weighting function defined by

$$w_\sigma(z) = ae^{-\frac{|z|^2}{\sigma^2}},$$

where a is a normalisation constant restricting the indicator function to range from 0 to 1. The variable σ is a scale parameter.

The ideal confidence map is defined as

$$c(x) = p(x|o) = be^{-\frac{|x-x^*|}{\alpha}^\beta}, \tag{2.1.2}$$

where b is a normalisation constant, α is a scale parameter, β is a shape parameter and x^* the object location. The confidence map is similar in principle to the voting space in Grabner, maximising and finding the peaks one

detects the target position. It is ideal because it gets generated in the first frame using the true starting position of the target. As time goes by the model updates the confidence map, and though the algorithm tries its best to maintain the ideal confidence map, it will never be ideal as it was during initialisation.

Having defined all the terms in Equation 2.1.1, the confidence map, context prior model and spatial-context model (filter), putting them together they formulate

$$\begin{aligned} c(x) &= p(x|o) \\ p(x|o) &= be^{-\frac{|x-x^*|}{\alpha}^\beta} \\ be^{-\frac{|x-x^*|}{\alpha}^\beta} &= \sum_{c(z)\in X^c} p(x|c(z),o)p(c(z)|o) \\ be^{-\frac{|x-x^*|}{\alpha}^\beta} &= \sum_{z\in\Omega_c(x^*)} h^{sc}(x-z)I(z)w_\sigma(z-x^*) \end{aligned} \tag{2.1.3}$$

Equation 2.1.3 is simply a convolution operation between the voting function (filter) and the context prior, giving our confidence map

$$be^{-\frac{|x-x^*|}{\alpha}^\beta} = h^{sc}(x) \otimes (I(x)w_\sigma(x-x^*)).$$

Moving this equation to the frequency domain, they get rid of the costly convolution operator and perform instead an efficient and fast multiplication giving

$$\mathcal{F}(be^{-\frac{|x-x^*|}{\alpha}^\beta}) = \mathcal{F}(h^{sc}(x)) \odot \mathcal{F}(I(x)w_\sigma(x-x^*)),$$

where $\mathcal{F}$ denotes the Fourier transform.

Now, putting all of the above into the context of a tracking algorithm, they propose the following. Given the target position is known at time t, they initialise the spatial-context model h^{sc} by dividing, in the frequency domain, the ideal confidence map defined in 2.1.2, with the image region around the known target position, including context, giving

$$h^{sc} = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(be^{-\frac{|x-x^*|}{\alpha}^\beta})}{\mathcal{F}(I(x)w_\sigma(x-x^*))}\right). \tag{2.1.4}$$

In frame $t+1$, where the target position is not known, they use the formula

$$c_{t+1}(x) = \mathcal{F}^{-1}\left(\mathcal{F}(H^{stc}_{t+1}(x)) \odot \mathcal{F}(I_{t+1}(x)w_{\sigma t}(x - x^*_t))\right) \tag{2.1.5}$$

to calculate the confidence map $c_{t+1}(x)$, using the spatio-temporal context model, $H^{stc}_{t+1}(x)$ and image region at frame $t+1$ centred around the last known position x_t (context prior). For the first iteration, the spatio-temporal context model, $H^{stc}_{t+1}(x)$ is set to the spatial-context model, h^{sc} calculated during the initialisation step.

The estimated target position is obtained by maximising the calculated confidence map

$$x^*_{t+1} = \underset{x \in \Omega_c(x^*_t)}{\operatorname{argmax}}\, c_{t+1}(x).$$

As the target moves, the region around the target and its context undergoes movement, rotation, scale variation and appearance changes. In order to still produce an accurate tracker in frame $t+2, t+3, ..., t+n$, capable of handling this change, the spatio-temporal context model H^{stc} in Equation 2.1.5, needs to be updated for the next frame and cannot continue to use the spatial-context model initialised and used in the first iteration, hence the term adaptive correlation filters.

The spatio-temporal context model for the next frame $t+2$ is updated as follows:

$$H^{stc}_{t+2} = (1-\rho)H^{stc}_{t+1} + \rho(h^{sc}_{t+1}),$$

where ρ is a learning parameter and h^{sc}_{t+1} the spatial-context model at frame $t+1$.

To obtain the most accurate estimation at time $t+2$, the updated spatio-temporal context model, H^{stc}_{t+2}, when applied in Equation 2.1.5, should yield something as close as possible to the confidence map defined in 2.1.2. To accomplish this, the spatial-context model h^{sc}_{t+1}, used when updating the model at the end of each iteration, is calculated using Equation 2.1.4.

This process, excluding the initialising phase, is repeated for each frame. The Figure in 2.5 from the article itself, is a good representation of the algorithm proposed.

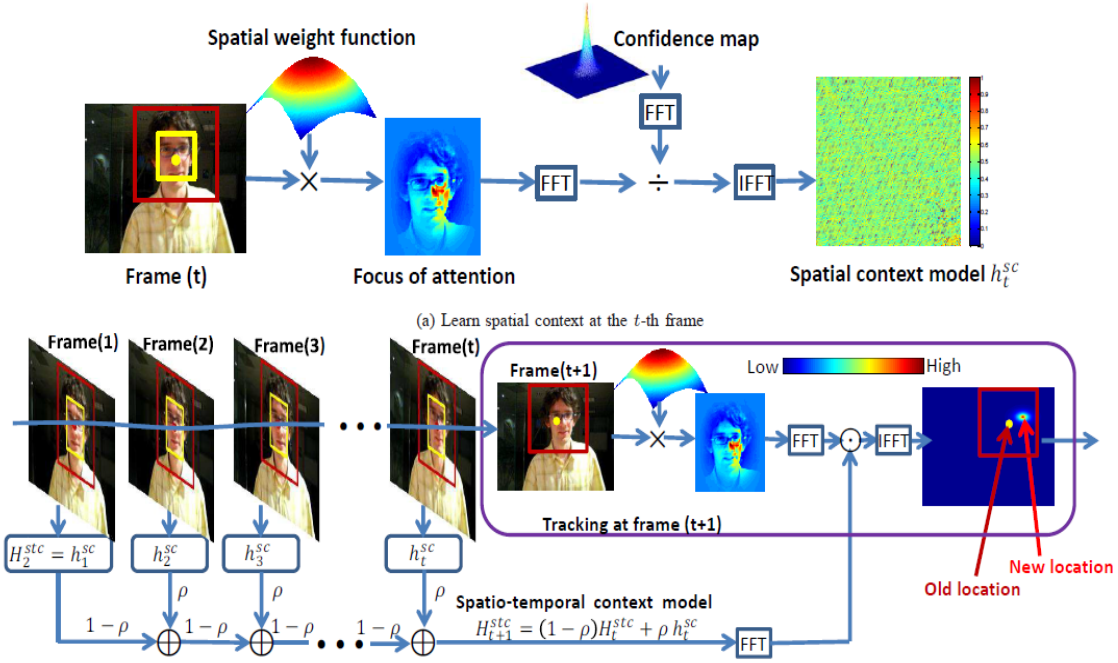


Figure 2.5: Fast tracking via spatio-temporal context learning

Their proposal although fast and efficient, also uses image regions as in Yang *et al.* [12], and only a subset around the target location. This limited area around the target is part of the reason the computation is so fast when using correlation filters. Expanding the area around the target decreases the speed of the tracker. Although their proposal can handle heavy occlusion, a simple test on the ***Red bull*** sequence in Section 4.2, shows it cannot handle full occlusion like Grabner.

A very powerful characteristic of their proposal is that no feature extraction is required. There is no need to find points of interest or perform segmentation to find clustered regions. The nature of correlation filters uses all the information from the target and surrounding area when convolution is applied. The most uneventful background to the target carries valuable information.

2.1.4 Object level context

In Li and Nevatia [15] the focus is on object level spatio-temporal relationships. These relationships are modelled using a dynamic Markov random

field (MRF) capable of both recognising and tracking simultaneously, similar to that of Yang *et al.* [12].

The integration of context is done in the following two ways:

- Spatial relationships (Co-inference between objects increase accuracy of model), and
- Temporal context (Accumulates object evidence and tracks continuously).

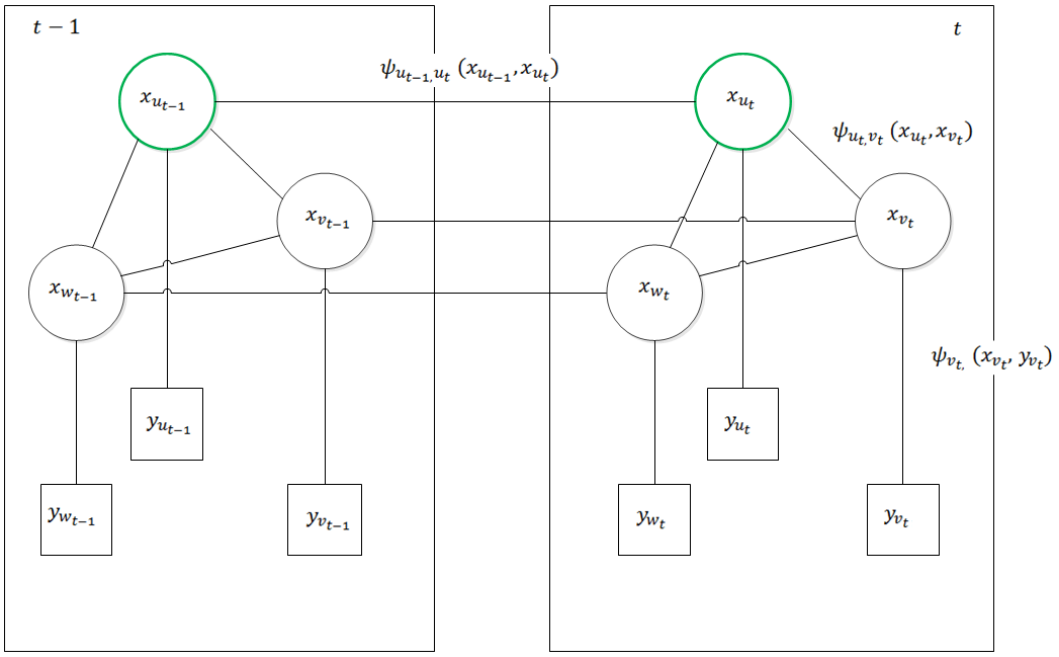


Figure 2.6: Markov random field

Looking at Figure 2.6, the concept of spatio-temporal modelling is well illustrated. The model has three types of edges, each associated with a potential function:

- Observation potential,
- Spatial potential, and
- Temporal potential.

Observation potentials are acquired from two sources. Firstly, to track humans in a typical meeting room they use a single-category object detector to detect humans giving potential $\psi_u(x_u, y_u)$, where x_u represents the unobserved state for object u and y_u the object evidence. Secondly, using a classifier based on the Bag of Features approach, they detect other objects of interest/context giving more observation potentials. An example of an observation potential for object v would be $\psi_v(x_v, y_v)$. The important thing to note is that any object recognition algorithm can be substituted here, with the main goal to detect categorised objects and use that as contextual information. The implementation of Hoiem *et al.* [1], as discussed in the next section, is a perfect candidate to substitute in here alongside an object detector.

The **spatial** potentials exists between objects of different categories, example of a spatial potential between object u and v is represented as $\psi_{v,u}(x_v, x_u)$. Unlike the implementation in Yang *et al.* [12], that only models the relationship between the context and target, they also model relationship between different types of context. The focus is on inter-category correlation between these objects. To use the example in Li and Nevatia [15], a person tends to sit on a chair beside a table, and a laptop is often near a person and on the table. Although the end goal is a spatial relationship, using information on how different categories interplay with one another falls into our category of informational modelling and can be extremely valuable when modelling spatial relationships and in a sense the essence of integrating context to its full extent.

The **temporal** potentials are tracked by means of optical flow from frame to frame. An example of such a potential for a single object u from time $t-1$ to t is represented as $\psi_{u_{t-1},u_t}(x_{u_{t-1}}, x_{u_t})$. As per the previous implementations, nodes get added dynamically to the model. To avoid false hypothesis from adding weak nodes, key nodes are detected first and provide contextual guidance for finding relevant objects.

Having a high level understanding of the three potential functions, the challenge is to build this model on-the-fly and to perform inference. These chal-

lenges are jointly resolved by belief propagation. The main reasons for using belief propagation to calculate inference, is the cycling and dynamic nature of the graph and inference should not be calculated over the entire sequence, but rather a sliding window. In the event of a cycling graph, loopy belief propagation is used unlike in Yang *et al.* [12] where belief propagation gave an exact estimate. Loopy belief propagation does not guarantee convergence, however it has proven excellent empirical performance.

Assuming the potential functions to be pairwise, the distribution of the MRF becomes

$$p(x,y) = \frac{1}{Z} \prod_{(v,u)\in\varepsilon} \psi_{v,u}(x_v, x_u) \prod_{v\in V} \psi_v(x_v, y_v)$$

where $\psi_{v,u}(x_v, x_u)$ represents the spatio-temporal relationship and $\psi_v(x_v, y_v)$ models the observation likelihood as described earlier. The proposed MRF is an undirected graph consisting of a set of nodes ν and edges ε. As mentioned earlier, each node $v \in \nu$ is associated with an unobserved state x_v and observation y_v. In a time sequence, a node at t and $t-1$ is represented as x_{v_t} and $x_{v_{t-1}}$ respectively. They define a node v at time step t as $x_{v_t} = (c_{v_t}, p_{v_t}, s_{v_t})$, where c_{v_t} is the object category label, p_{v_t} the coordinates of the centroid and s_{v_t} the logarithm of size. The object evidence for x_{v_t} is defined as y_{v_t}. The more categories of objects they model, the more sets of nodes get added, that is $w \in W$.

Unlike the previous implementations, this algorithm uses an object detector to obtain object level context. Advantages of modelling at this level means fewer objects reducing the computations and complexity of inference. Real-world objects are far less prone to occlusion and appearance changes compared to low-level features. They also provide far more stable spatio-temporal relationships.

The model they propose are heavily dependent on object categories; the more categories that can be recognised, the more context can be used. Knowledge of the interplay between different categories is also required. All of these require upfront investment in some form or another, that is, training more classifiers and learning interplay between categories.

2.2 Informational modelling

Context itself contains more information than just position and motion to other objects. This additional information can be invaluable when designing a model, giving us certain assumptions, reducing uncertainties, and improving the final outcome of a model. Co-occurrence between objects does not only give motion correlation, but also the knowledge that a group of items tend to coexist. Having knowledge of the ones existence gives assumptions or verification of the other.

Hoiem *et al.* [1] is a good example where visual context serves as knowledge instead of only focusing on the positional properties. As highlighted in the beginning of this chapter, their implementation is not focussed on tracking, but rather object detection. The importance is to capture the interplay between objects for better scene understanding. They have built a framework to place any choice of object detector within context of an overall 3D scene by modelling the interdependencies between objects, surface orientation and camera viewpoint. The two key outcomes of their approach are

- 3D reasoning improves object detection, and
- The more complete the scene is modelled, the more objects and context, the better the estimates are.

They use context information to model a better 3D scene understanding and use that information in the task of object detection. Their model consists of the following three elements

- Low-level object detectors,
- Rough 3D scene geometry, and
- Approximate camera position and orientation.

Perspective projection from 3D to 2D loses valuable information and to use the example in the article, a nearby car looks different from a car far away

when projected onto the 2D plane. The best way to explain the benefits of scene understanding is Figure 2.7 taken from Hoiem *et al.* [1].

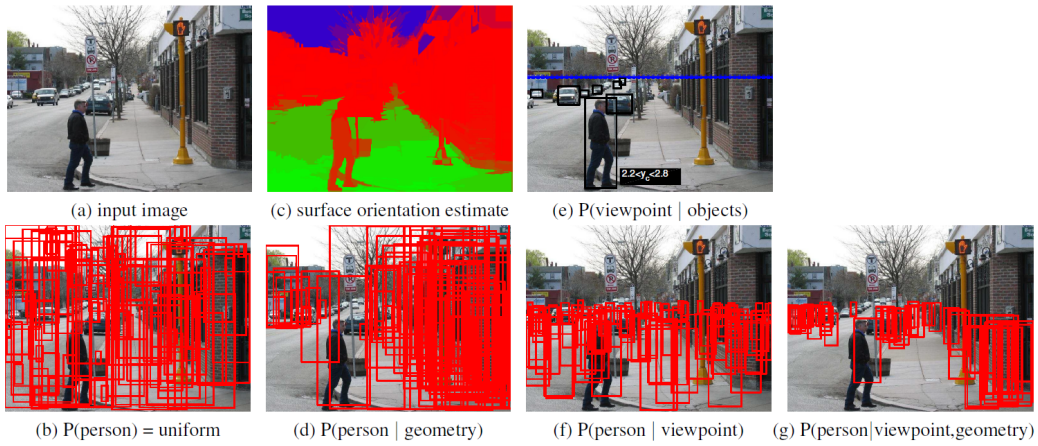


Figure 2.7: Putting things in perspective, Hoiem *et al.* [1]

In the typical 2D scene as in Figure 2.7, the likelihood of finding a pedestrian at any position in the image under any scale is equally likely as indicated in Figure 2.7b leading to the problem of position and scale. Hoiem *et al.* [1] resolves the problem of position by estimating and modelling the rough surface geometry of the scene. Knowing where the road, buildings and sky are, one can reduce the uncertainty of where pedestrians potentially can be, see Figure 2.7d.

Modelling the camera viewpoint provides a likely scale of objects in the image and helps with the issue of scale. For example, large pedestrians in the far end of the street are not likely, unless the person is as tall as a grown tree, see Figure 2.7f. Combining $p(pedestrian|geometry)$ and $p(pedestrian|viewpoint)$ giving $p(pedestrian|geometry, viewpoint)$, we have a much richer understanding of the scene which gives us a solid prior likelihood for the position and scale of a pedestrian. The next thing is to apply the object detector for pedestrians given this prior.

The objectives of their model is to determine viewpoint, object identities and surface geometry from a given image. Although these three elements can be modelled independently, understanding their scene interactions makes

for a more accurate estimation. The interaction between these elements is represented as a graphical model in Figure 2.8.

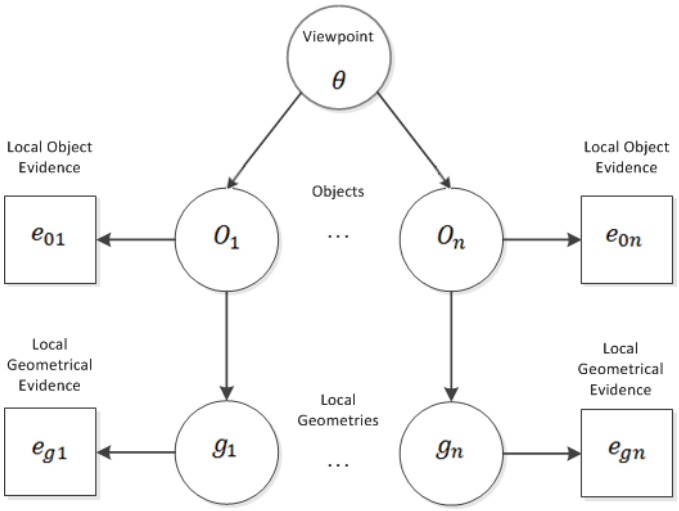


Figure 2.8: Perspective model

The key information to take away from this model is that the object position and size depends on the viewpoint θ, and surface geometry g depends on the object identity o. The conditional independence is that the surface geometry is independent of viewpoint given object identity $P(g|\theta, o) = p(g|o)$, and object identity is independent of surface geometry given viewpoint $p(o|\theta, g) = p(o|\theta)$. Given image evidence for surface geometry e_g and object identity e_o, the model can be written as

$$P(\theta, \mathbf{o}, \mathbf{g}|\mathbf{e}) \propto P(\theta) \prod P(o_i|\theta) \frac{P(o_i|e_0)}{P(o_i)} P(g_i|o_i) \frac{P(g_i|e_g)}{P(g_i)}$$

where $e = \{e_g, e_o\}$, $P(o_i|\theta)\frac{P(o_i|e_0)}{P(o_i)}$ represents the object model and $P(g_i|o_i)\frac{P(g_i|e_g)}{P(g_i)}$ the surface geometry.

The consideration of both surface geometry and viewpoints, especially viewpoints, improved object detection. Given the positive results of viewpoints, Hoiem *et al.* [1] concluded that the more types of objects than can be identified, the better the horizon estimates will be, leading to a better object detector. Object and geometry evidence can also improve horizon estimates even though they are separate from viewpoint.

2.3 Summary

Outlining what was presented in the background study, we started with the simple but highly effective supporter model implemented in Grabner. This implementation was a pure spatio-temporal model between context and target with emphasis on co-occurrence. They used low-level uncategorised features for context, local feature points using a Harris point detector decoded using SIFT descriptors to be precise. These local points can be plentiful and having so many candidates for context, it is important making sure no unnecessary context gets modelled, resulting in computational strain and polluted estimates. Local feature points do not capture object structure information that well and suffers from proper real-world characteristics with predictable movement. They are also highly prone to occlusion and appearance changes. However detecting matches for these features in subsequent frames are simple and having plenty of them, the odd occluded feature point has little impact.

Next we looked at the Context-Aware Tracker (CAT) in Yang *et al.* [12], where context is represented as image regions. These regions are still low-level, in that it does not necessarily have real-world predictable behaviour, however they are less prone to occlusion and appearance changes. The regions are tracked using a Mean-shift algorithm and those that frequently co-occur with the target are chosen as candidates. Subspace analysis of these candidates are performed to estimate the affine motion model between target and candidate. Only the candidates that meet the necessary criteria for good correlation with the target is used in the model. The spatial relationship between target and context is calculated as a by-product from the subspace analysis. All these potentials between target and context gives a dynamic Markov random field with star topology. A belief propagation algorithm over the MRF gives an exact estimate for where the target position is.

In Zhang *et al.* [14] we discussed a model with a similar Bayesian framework as in Grabner. They implemented a fast and efficient algorithm using adaptive correlation filters capable of tracking under heavy occlusion. Context

is taken as a subset region around the target location. A powerful characteristic of their proposal is that no feature extraction is required. There is no need to find points of interest or perform segmentation to find clustered regions. The model however is closely tied to how correlation works and does not give room for much extension or generalisation. To include more context, the region around the target needs to be extended, but comes at a computational price.

In Li and Nevatia [15], context is chosen at a real-world object level. Pre-processing is performed to identify a set of objects that the model understands. Since objects are far less in quantity than local feature points, and the fact that they have predictable movement, ensures for a more stable model. The implementation also used a dynamic Markov random field that models both temporal and spatial potentials. Additionally relationships between different types of context also gets modelled, unlike in Yang *et al.* [12] and Grabner, where the relationships were only between target and context. In this implementation belief propagation is also used to infer the model estimates.

In all three of the following implementations Grabner, Yang *et al.* [12] and Li and Nevatia [15], the model is learned when the target is visible and can be used as a verification mechanism to the main target tracker. All these models leave room for the implementation of any tracker/detector. When the target tracker no longer finds a visible target, their models are capable of finding the target.

The last article Hoiem *et al.* [1] focuses on the object detection phase, a component of a typical end-to-end tracking algorithm. This model performs no spatial or temporal modelling and purely uses the context as knowledge reducing the uncertainty when detecting an object. This implementation also requires real-world objects to be detected and requires pre-processing.

It is abundantly clear that context is invaluable to the tracking algorithm. The challenge lies with the nature of the context we choose. Low-level uncategorised context gives room for a generic algorithm not requiring a

focused understanding of a specific problem. As the context gets more abstracted to the point where it is real-world objects, additional pre-processing is required, and also one has to know the class of objects to expect. That is, if you are expecting to track a person in a typical office environment, you can make assumptions and expect certain objects like laptops, chairs, notepads, and so forth to be present. Should your tracker suddenly need to track a person in a crowded shopping mall, the algorithm needs updating in what objects to look for as context. On the other hand modelling is less computational and reliable. Integrating the knowledge of interplay between context and using real-world objects improves the reliability of the tracker, but we loose generalisation.

Chapter 3

Supporter Model

3.1 Overview

This chapter provides more detail of what was discussed briefly in the overview regarding the supporter model of Grabner. It covers how this model was improved upon in this work to overcome some of the challenges when tracking with context. It also covers the integration of the work done in Alexe *et al.* [10] into the model as a potential real-world class-less supporter model. Going forward it is referred to as the Objectness supporters.

The most important feature about the supporter model as one will see from this chapter, is how simple, yet highly effective it is. In the broadest term, supporters move temporarily with the object in such a way that we can predict the position of the tracked object under abrupt appearance changes as well as full occlusions. As mentioned in Section 2.1.1, the supporter model is inspired by the Implicit Shape Model (ISM) algorithm (Leibe *et al.* [16]) where local image features (context) vote for the object position. At this point it is worthwhile to discuss the ISM algorithm as an object class detector and how the voting mechanism is implemented.

3.1.1 Implicit shape model

The two main components of the ISM object detector are the appearance codebook and the occurrence distribution of the codebook entries. The ap-

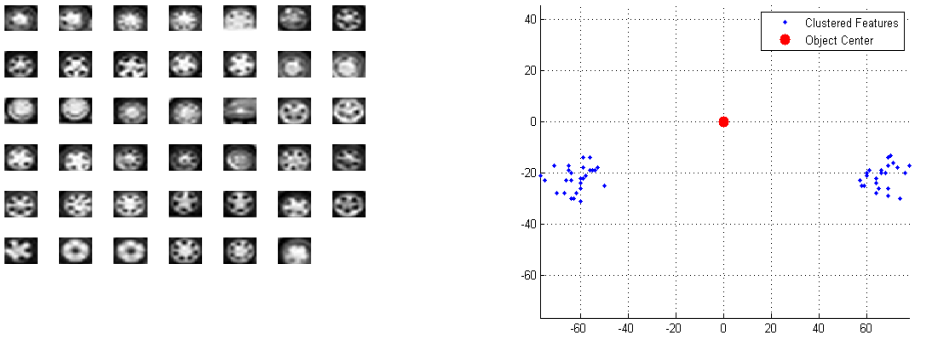
pearance codebook is obtained by processing samples for a given object class. Given a set of images for objects of the same type, it extracts local image features using a basic point detector such as the Harris corner detector. It encodes these feature points as SIFT descriptors and applies a K-means clustering algorithm in the descriptor space, grouping these to create the appearance codebook entries. Each cluster represents a single codebook entry and a specific feature.

The next important component is to calculate the corresponding occurrence distribution for each code book entry (feature). It should be noted that a stand-alone object is represented by multiple codebook entries. The occurrence distribution for a single codebook entry is a set of relative locations that indicates where this entry has been observed relative to the object centre. Occurrences are given as relative to the object centre and not as absolute coordinates. It is the relative measurements that make the ISM representation of an object invariant to translation.

Explaining the two components above is best done using an example. Given a training set of cars from the side, wheels have similar local feature points irrespective of being the front or back wheel. Applying the K-means clustering algorithm over all these features extracted from the set of cars, it will cluster all features part of the wheels together as a single codebook entry (cluster centroid), bearing no position in the image. The occurrence distribution for the cluster of wheels (codebook entry) will have two relative locations, one left to the centre of the car and one to the right. The wheels are only one codebook entry with occurrence distribution. During training the K-means algorithm groups more clusters of features for the same set of training data, resulting in many codebook entries and occurrence distributions that define the object.

Figure 3.1a shows a cluster of wheels from the training set, along with its occurrence distribution in Figure 3.1b

The key implementation of ISM as an object detector and specifically the part that inspired the algorithm in Grabner, is how the model detects an

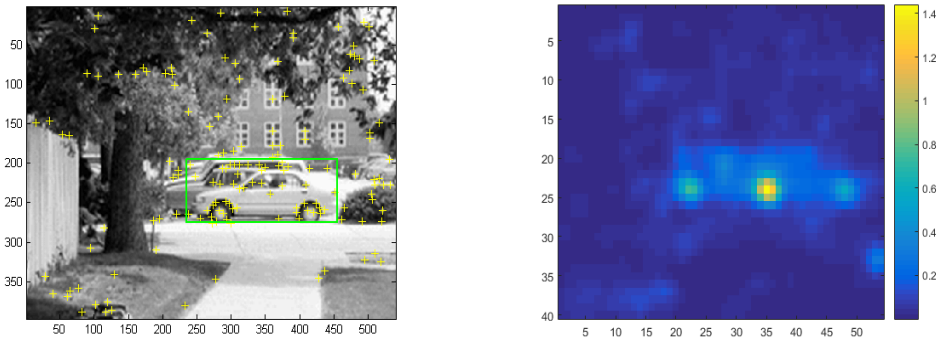


(a) Code book entry - cluster of wheels (b) ISM voting - accumulator space

Figure 3.1: ISM training step

object of the same class using the appearance codebook and occurrence distribution constructed during training. Given a test image, it starts by extracting feature points using a similar or different point detector used during training. These feature points get encoded as SIFT descriptors and are then matched against the appearance codebook entries. For every codebook entry that gets activated, we use all sets of relative positions in the occurrence distribution of that codebook entry to vote using a method based on the Generalised Hough Transform. Voting occurs in the Hough parameter space also called the accumulator space. See Figure 3.2. We end with a complete distribution across the image space where the target potentially might be found. Using the position of the highest peak in the two-dimensional distribution provides the object position. Figure 3.2 illustrates how ISM detects a car along with the voting distribution in the accumulator space for the given image.

The main difference between the supporter model and ISM, and what makes it so complementary to our goals, is that the appearance codebook and occurrence distribution entries for supporters are built on-the-fly. It requires no upfront training. If the motion of a supporter to the target becomes uncorrelated, that supporter is no longer used in the voting. This is one of the key advantages of this model, its ability to adapt to change.



(a) ISM voting - detecting Car

(b) ISM voting - accumulator space

Figure 3.2: ISM voting step

3.2 Improved supporter model

Referring back to the objectives set in Section 1.5, one of the key goals is a generalised tracking model, capable of learning on-the-fly from unlabelled data, and tracking in an unconstrained environment. In Chapter 2 we showed the benefits of context and how it improved the ability of tracking in these environments. Low-level supporters are much more prone to occlusion and appearance changes than real-world objects. Using real-world objects for supporters ensures for more predictability, reducing the uncertainty when integrating information into the model. The number of real-world objects is far less in a single image than a large number of low-level features. This results in a much smaller and controlled set to manage/update and reduces the chance of unwanted supporters polluting the model, being only present for a small period and then disappearing. Beneficial supporters are those that share consistent movement with the target and are most likely to be present in the next frame.

However, the model is not so general when we use class specific context. The most important benefit a generalised tracker can receive from context is the temporary co-occurrence of context with the target, irrespective of the nature of the context. Class-less object detectors as per Alexe *et al.* [10] work well in unconstrained environments, but might not find that many relevant objects due to their generalised nature.

We propose a context flexible model that can deal with different types of context. We based our model on the work done by Grabner, prioritising the integration of real-world predictable and reliable context using off-the-shelf object detectors. Based on the availability of objects, it has the option to fall back on the low-level uncategorised context. This hybrid approach ensures for a generalised tracker applying higher-level scene understanding to improve its effectiveness. To maintain generalisation, the extent to which we capitalise on higher-level scene understanding comes down to co-occurrence with the target and focusing only on the positional benefits of integrating context, and not so much the knowledge of interplay between different types of objects and how they behave in a particular scenario. Focusing on the latter reduces generalisation of our tracker.

3.3 The model

Beginning with the most basic definition of what we want, our aim is to learn a distribution $p(x|I_t)$, predicting the object position x in image I at timestamp t. Using conditional probability we have

$$p(x|I_t) = \frac{p(x, I_t)}{p(I_t)}.$$

Our goal is to add context to derive a more usable model. If we integrate the context available in the image into our equation using marginalisation we have

$$p(x|I_t) = \frac{\sum_{\mathbf{s}\in S} p(x, I_t, s_1, .., s_i)}{p(I_t)}$$

where $\mathbf{s} = s_1, .., s_i \in S$ is the set of supporters in the image I_t. The term $p(I_t)$ is independent of x. We are only interested in proportionality, hence we can get rid of the term $p(I_t)$ beneath the line and simplify the equation to

$$p(x|I_t) \propto \sum_{s\in S} p(x, I_t, s_1, .., s_i).$$

Replacing $s_1, .., s_i$ with the vector $\mathbf{s}$ gives us a more readable equation

$$p(x|I_t) \propto \sum_{s\in S} p(x, I_t, \mathbf{s}).$$

Using the product rule recursively we have

$$\begin{aligned} p(x|I_t) &\propto \sum_{s \in S} p(x|\mathbf{s}, I_t) p(\mathbf{s}, I_t), \\ &= \sum_{\mathbf{s} \in S} p(x|\mathbf{s}, I_t) p(\mathbf{s}|I_t) p(I_t). \end{aligned}$$

Marginalising over $\mathbf{s}$, $p(I_t)$ is a constant and adds no contribution to the final outcome giving,

$$p(x|I_t) \propto \sum_{s \in S} p(x|\mathbf{s}, I_t) p(\mathbf{s}|I_t).$$

If we assume position x is independent from image I_t given features $\mathbf{s}$, we have

$$p(x|I_t) \propto \sum_{s \in S} p(x|\mathbf{s}) p(\mathbf{s}|I_t). \tag{3.3.1}$$

To interpret $p(x|I_t)$, we need to remember that x is defined as the object position in a two-dimensional image. In Section 3.4.5 we describe in detail exactly how this two-dimensional distribution gets inferred. In short, our distribution x is dependent on the image size, this makes it computationally intensive and less likely to find definitive peaks. A method based on the Generalised Hough Transform is used to infer the object position in a much smaller controlled distribution that will guarantee peaks by reducing the two-dimensional size. Mapping the coordinates of the maximum peak back to the image space/coordinates provides the target location. The estimated position might be out by a few pixels as a result of mapping, but this is negligible.

Understanding the output and goal of our model, the next challenge is to state how $p(\mathbf{s}|I_t)$ and $p(x|\mathbf{s})$ are calculated. The distribution $p(\mathbf{s}|I_t)$ represents the indicator function, indicating the presence of a supporter in the image and can be seen as a sharply peaked delta function having two outcomes, zero and one. A supporter is either present or not. The distribution $p(x|\mathbf{s})$ represents the voting function that contributes to the voting space $p(x|I_t)$, given the indicator function considers the supporter found.

3.4 Algorithm

This section explains the entire algorithm, how we learn our model on-the-fly and then apply the model in the event of occlusion or sudden appearance changes to the point where the target is not trackable. Specific changes to the model in Grabner will be highlighted in the sections below. The algorithm illustrates how the model is learned and used in the typical tracking loop.

Algorithm 1 Learning and applying the model

```
1:  Instantiating the model
2:  - instantiate the main tracker & supporter model (Section 3.4.1)
3:  while run do
4:      Managing supporters
5:      - detect or track supporters (Section 3.4.2)
6:      - detect or track target
7:      - verify target position (Section 3.4.3)
8:      if target is found then
9:          Learning the model
10:         - update or add supporter (Section 3.4.4)
11:     else
12:         Applying the model
13:         - find target using model (Section 3.4.5)
14:         if target position confident then
15:             - build second-level supporters (Section 3.4.7)
16:         end if
17:     end if
18: end while
```

3.4.1 Instantiating the model

As illustrated in Figure 1.1, the typical tracking algorithm has a basic loop to accomplish its goal. The supporter model requires a few basic parameters to be set before learning can begin. This is done outside of the normal tracking loop and is also the place where any off-the-shelf tracker can perform its initialisation.

3.4.2 Managing supporters

In this step it is about estimating the indicator function $p(\mathbf{s}|I_t)$, indicating the presence of supporters in the image. In the most general description, whatever object detector or tracker we use to find potential supporters in subsequent frames, we need to determine whether the supporters $\boldsymbol{s}$ found in the current image I_t already exist in the database/model. If a supporter is

found, the current position of the supporter is updated in the database and the supporter gets marked as active. If the supporter could not be found in the database, it is placed in a separate list of new supporters, **l**. At this stage no learning is performed. The information gathered will be used at a later stage when either learning or applying the model.

We define a supporter as $s_t = (d_t, p_t, r_t, \vartheta_t, \Sigma_t)$, where the subscript t represents the current time, d_t represents the identifying data to match against in the database, p_t the position of the supporter, r_t the radius (distance) between supporter and target, ϑ_t the angle between supporter and target, and Σ_t the covariance matrix of this supporter relative to the target for the two variables r_t and ϑ_t.

The basic implementation of this model uses SIFT features for the supporter data d. The data for a single SIFT descriptor consists of a 128 dimensional vector consisting of 8 bin orientation histograms extracted around a total of 16 neighbouring patches around the local point of interest. Matching in the supporter database is done using a dot-product of the normalised SIFT descriptors. If above a set threshold θ, we consider it a match. A single descriptor can match multiple entries in the supporter database, so we take the match with the highest score.

$$max_d \in DB(s^T \cdot d) > \theta,$$

where d represents the SIFT descriptor being matched.

This simple and effective mechanism for matching SIFT supporters rather accurately, is extremely useful especially when there is abrupt movement, like a sudden change in camera position. The matching mechanism is nothing more than object detection, and with abrupt movement, detection is more reliable than tracking. In the original implementation the use of a KLT tracker is suggested in order to ensure that false supporter matching is reduced from frame to frame. If a given situation can guarantee smooth movement, then tracking of supporters can definitely benefit the accuracy of the model, however we found that the slightest movement of the camera results in the KLT tracker losing its effectiveness.

Focusing the task of finding and matching supporters strictly a detection task and not a tracking task, makes for a more robust model capable of handling changes in movement. The challenge is to ensure that mismatches during detection do not affect the outcome of the voting too aggressively. This concern is somewhat reduced given the model depends on temporary links and its ability to ignore supporters drifting. To further reduce the uncertainty of a mismatch, we improve the model by introducing two mechanisms below in Section 3.4.2.1 and 3.4.2.2.

To introduce higher-level scene understanding, the implementation of our model uses SIFT features as default supporters but gives the option of using, along with SIFT supporters, any off-the-shelf object detector or tracker for finding supporters at a real-world object level. A separate model is created, managed and updated for each unique object detector/tracker implementation. If a selected object detector is capable of detecting a range of classes, a single model will be maintained for all the possible classes of objects it can detect. In the same way that SIFT is our default low-level supporter model, we also want a default real-world supporter model. This ensures for a stand-alone implementation that provides basic low-level and real-world level supporters, with the option to add class-specific supporters as needed. In Section 3.4.8 on Page 52 we briefly explain the essence of the work done in Alexe *et al.* [10] and how we integrated it with our model as a class-less object detector.

The solution allows for any number of models to be maintained. A weighting for each unique model should be defined to decide how much each model should contribute to the final result. Further details on the weighted voting mechanism can be found in Section 3.4.5.

3.4.2.1 Relative awareness

This check is optional to the primary matching technique for each off-the-shelf implementation when matching against the database. The benefits of context does not have to be a one directional benefit. Supporters should also benefit from the relationship with the target. Relative awareness involves

calculating the relative radius and angle between candidate supporter and estimated target position using Equation 3.4.1. These values are matched against the last recorded relative radius and angle for each supporter in the database. If the relative match is within tolerance it is considered a relative match.

To provide a target estimate that can be used in our relative awareness mechanism, we sacrifice the SIFT supporter model to provide an estimate. The SIFT supporters themselves therefore cannot benefit from relative awareness; all other potential supporter models can. We chose SIFT due to its effective matching ability using the dot-product and the high accuracy it provides. This requires SIFT features to always be trained, even if the weighting is set to zero for the main voting purpose. In Section 4.6 on Page 82 we show the value of relative awareness. In the next section on stale supporters we show another mechanism, one that even SIFT supporters can benefit from.

3.4.2.2 Stale supporters

To maintain a proper supporter model, one has to get rid of stale supporters. A stale supporter is defined as a supporter that has not been detected in n subsequent frames. Having stale supporters affects the number of elements to match against, increasing the chance of a false positive match or potentially using an outdated supporter. The benefits of getting rid of stale supporters are illustrated in Section 4.7 on Page 83 with a basic experiment.

3.4.3 Verification

As mentioned in the problem statement, (Section 1.3), generative models are vulnerable to drift due to their online-updating capabilities. Kalman filters use observations over time to correct the drift of the dynamic equation. Having some form of verification mechanism to improve the trustworthiness of our model estimate will help us reach our objective as set out in Section 1.5.

In Section 2.1.1, the model proposed by Grabner does not mention or utilise the verification potential of their model. In Yang *et al.* [12], the verification

properties of context are integral to their solution and something we will incorporate into our model as well. To benefit from the verification properties of the model, we need only to apply the model. At this stage we will assume we have a trained model. Details of learning and applying the model follow in the sections below.

In our implementation we ensure that our model has at least trained on a configurable number of frames before trusting any verification contribution. If none is provided we use a default of ten. With a trained model and knowledge of all the active supporters in the current frame, we apply the model to determine where we think the target should be. Using the estimated position from verification, along with the position estimated by the main target tracker, we can determine whether the main tracker is drifting to a false positive. We accomplish this by verifying if both positions are within a configurable range (tolerance) from each other.

In Section 4.2.2 on Page 56 we detail two experiments, ***Pedestrian*** and ***FaceChange***, that show how the verification mechanism avoided drifting in the main target tracker. See Figures 4.6 on Page 62 and 4.10 on Page 64.

3.4.4 Learning the model

If the main tracker or detector finds the target we use the opportunity to learn our model, $p(x|s)$ specified in Equation 3.3.1. In Grabner they only learn the model when either the target or one of the supporters move. We propose learning on every iteration even when nothing moves in two subsequent images. Information still exists when nothing moves. It shows the relationship between supporters and the target is still intact, strengthening that link.

Learning starts by updating all supporters already in the database that got marked as active in the detect and tracking phase. Updating an existing supporter (i) involves calculating the radius, angle and covariance matrix

relative to the target in the current frame using

$$\begin{aligned} r^{(i)} &= \|x_t^{(i)} - x_t^*\|_2 \\ \vartheta^{(i)} &= \angle(x_t^{(i)}, x_t^*) \\ \Sigma^{(i)} &= (x_t^* - \mu^{(i)})(x_t^* - \mu^{(i)})^T \end{aligned} \tag{3.4.1}$$

where x_t^* represents the target position and $\mu = \begin{bmatrix} r \\ \vartheta \end{bmatrix}$ using the average radius r and angle ϑ. To benefit from the temporal information between frames and for supporters to earn their worth, we apply the exponential forgetting principle when finalising the parameters using

$$\begin{aligned} r_t^{(i)} &= \alpha r_{t-1}^{(i)} + (1-\alpha) r^{(i)} \\ \vartheta_t^{(i)} &= \alpha \vartheta_{t-1}^{(i)} + (1-\alpha)(\vartheta^{(i)} - \vartheta_0^{(i)}) \\ \Sigma_t^{(i)} &= \alpha \Sigma_{t-1}^{(i)} + (1-\alpha)\Sigma^{(i)} \end{aligned}$$

where α is a value between zero and one, specified when the model gets initialised.

Once the existing supporters have been updated we add all new supporters in the list $\boldsymbol{l}$ to the database and initialise them. Initialising a supporter consists of setting the current position p_t and calculating the radius r_t and angle ϑ_t relative to the target as per Equation 3.4.1. The covariance matrix Σ however gets initialised with a default value of

$$\Sigma = \sigma I_2$$

where σ is equal to the square of the distance between supporter and target, and I_2 is the 2x2 identity matrix. This forces a supporter to undergo proper training to earn its place in the model. It also gives preference to supporters closer to the target. The further from the target the harder it has to train to earn trust. At the risk of self-learning we also multiply σ with a factor of two when we initialise second-level supporters.

The reason for keeping new supporters in a separate list until this point is to avoid the situation where the target is not located and we need to apply our model. In that event new supporters in the list $\boldsymbol{l}$ will not get the opportunity to be initialised and trained as first-level supporters, but will rather

be considered as second-level supporters. See Section 3.4.7 for details on second-level supporters.

3.4.5 Applying the model

When the main tracker or object detector fails to find the target, the supporter model gets applied in order to locate the target. At this stage we have a database of potential active supporters as well as a list of new supporters. Applying the model consists of using the voting function $p(x|\mathbf{s})$ for each active supporter and to estimate the voting function $p(x|I_t)$.

In Section 3.1.1 we explained that the supporter model voting principles are very similar to the way the ISM algorithm detects an object, using the appearance codebook and occurrence distribution. In this instance our appearance codebook is the active supporters and the occurrence distribution is the voting function learned over time. Voting for the target position is accomplished by performing the following steps for each active supporter, including second-level supporters:

1. Obtain the voting value for this supporter using

$$P(x|s) \propto \frac{1}{\sqrt{2\pi|\Sigma|}} \exp(-0.5(x-\mu)^T\Sigma^{-1}(x-\mu)) \tag{3.4.2}$$

   where x represents all radii and angles relative to this supporter and $\mu = \begin{bmatrix} r \\ \vartheta \end{bmatrix}$ using the last observed relative radius r and angle ϑ for this supporter pointing to the target. Instead of setting μ to the average relative radius and angle for this supporter we focus the maximum voting contribution for the target using the last observed radius and angle given the observed position for this supporter. The variable Σ represents the covariance matrix for this supporter learned over time for variables r and ϑ. The more motion correlation a supporter shares with the target the higher the voting contribution.

2. Apply the priority filter explained in Section 3.4.6 to obtain the final voting value.

3. Calculate the target position using the supporter's current position and its relative position to the target, using the last observed radius and angle from training.

4. Map the target position from the previous step to the Generalised Hough Transform space and aggregate the voting value calculated in Step 2 to that cell.

Repeating these steps for all the active supporters results in a two-dimensional distribution function. This voting space is best illustrated in Figure 2.2.

The target position is obtained by finding the maximum cell and mapping the coordinate from the Generalised Hough Transform space back to the actual image dimensions.

This explains the voting process for a single supporter model. As indicated earlier in this chapter our solution allows for any number of models to contribute to the voting space. This is accomplished using a unique weighting for each model when calculating the final result using

$$p(\mathbf{s}|I_t) = \sum_{i=1}^{n} \omega_i p(x|\mathbf{s})_i,$$

where n represents the number of unique supporter models, $p(x|\mathbf{s})_i$ the voting contribution from the i-th supporter model and ω_i the weighting for the i-th model set during the initialisation phase at the start of the algorithm.

3.4.6 Priority filter

The importance of correlated movement with the target has been expressed on multiple occasions throughout this thesis. As highlighted in Grabner, experiments showed that having few but accurate information sources leads to better results. The focus should be on quality supporters above quantity.

During experimentation the importance of the above observation was highlighted time and again, especially during experimentation on the strength

of second-level supporters. We found that although we still had good trustworthy first-level supporters, the second-level supporters trained quickly caused us to drift from the target. It is important to prioritise the good correlated first-level supporters for as long as you can.

We needed a mechanism that can accomplish the following two criteria:

- prioritise good correlated supporters over weaker ones,
- prioritise first-level supporters with good correlation over that of second-level supporters.

We accomplished both these criteria by applying a global multivariate Gaussian filter to each active supporter prior to casting its final vote. The multivariate Gaussian is represented by two variables, namely radius and angle standard deviation. Each supporter that needs to vote provides as input to the multivariate Gaussian its radius and angle deviation. The value returned is used as a multiplier/filter prior to voting in the Generalised Hough Transform space. We have two Gaussian filters, X_1 for the first-level supporters and X_2 for the second-level supporters, represented as follows:

$$X_1 \sim \mathcal{N}(\mu, \Sigma)$$

$$X_2 \sim \frac{1}{2}\mathcal{N}(\mu, \Sigma)$$

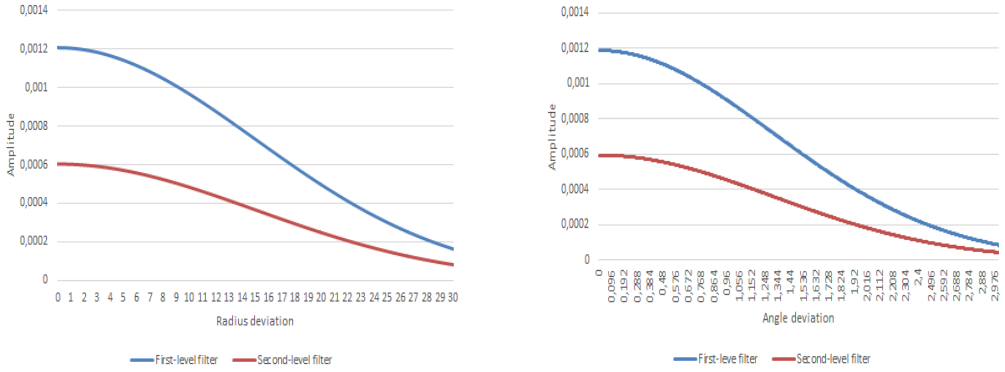
with μ set around zero and

$$\Sigma = \begin{bmatrix} {\sigma_r}^2 & 0 \\ 0 & {\sigma_\vartheta}^2 \end{bmatrix}$$

where $\sigma_r = 15$ represents the standard deviation for the radius and $\sigma_\vartheta = 1.309$ (75 degrees), the standard deviation for the angle. These particular values were chosen to ensure that any standard deviation for the radius and angle above these values will fall outside the 68.27% contribution away from the mean of the multivariate Gaussian. Experimental results has shown that these values work well. The voting contribution from good supporters gets further emphasised and reduces the contribution from weak

supporters, eliminating potential noise in the final estimate. The parameters for this filter is configurable.

The nature of the normal distribution allows for the first criteria to be met. The closer the standard deviation is to zero, the higher the multiplier. Having a second Gaussian filter with half of the amplitude of the first for the second-level supporters, ensures our second criteria is met. Figure 3.3 illustrates the filter values for first and second-level around the origin for both radius and angle deviation.



(a) Radius deviation (b) Angle deviation

Figure 3.3: First and second-level priority filter

Results on the effectiveness of our filter can be found in Section 4.5 on Page 81.

3.4.7 Second-level supporters

In Section 2.1.2 for the CAT implementation, when the target is assumed drifting or invisible, the mining process gets suspended. In Grabner a second-level supporter model gets initialised and learned, using context even when the target tracker fails.

Second-level supporters are not much different from first-level supporters in how we update and apply them. When the target is not visible and first-level supporters get applied, no updates on the model are allowed. Instead a temporary second-level supporter model is built for the duration the target is not visible. As soon as the target is visible again, the second-level

supporter model gets discarded. Initialising a second-level supporter is done exactly like a first-level supporter, except that the covariance matrix is multiplied by a factor of two, to reduce the effects of self-prediction.

In Section 4.4 on Page 76 we show the value of second-level supporters by training for a maximum of five frames and then disabling the main target tracker. This way we force our model to continue tracking until first-level supporters thin out and second-level supporters carry on.

3.4.8 Real-world supporters

Now that we have covered the main algorithm of our model, we focus on the integration of Alexe *et al.* [10] as a potential real-world supporter model into the proposed model. We discuss the potential it holds, and the challenges we faced during integration and experimentation.

We indicated in Section 3.4.2 that our model can take any number of object detectors to find potential real-world supporters. We chose the implementation of Alexe *et al.* [10] as our default real-world supporters purely to provide a standalone implementation. One simply needs to provide a target tracker of choice and our model can be trained and applied. To further improve the certainty of the model in particular scenarios, the use of class-specific supporter models can be integrated as needed.

The implementation in Alexe *et al.* [10] quantifies the likelihood for a box within an image to contain an object, irrespective of class. Several cues used in multiple possible permutations are combined in a Bayesian framework to provide a likelihood score for each possible box detected. Experiments shows that combining cues outperforms the use of a single stand-alone cue. The cues presented in their work are:

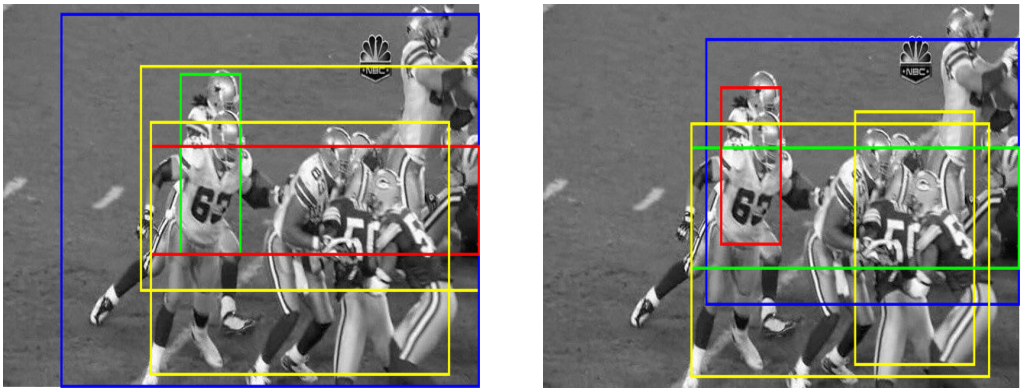
- **Multi-scale saliency (MS)** - Uses a saliency measure based on the spectral residual of the FFT.
- **Colour contrast (CC)** - Measures the dissimilarity of a box with its immediate surrounding area.

- **Edge density (ED)**- Measures the density of edges near the box borders.
- **Superpixels straddling (SS)** - Capturing closed boundary characteristic of objects using superpixels as features. Superpixels are areas in an image of similar color or texture.
- **Location and size (LS)** - Assess the likelihood of a box containing an object given its location and size.

Experimentation has shown that the following combination, Multi-scale saliency, Colour contrast and Superpixels straddling (MS+CC+SS), provided the best complementary set of cues. This is also the default combination we use in our implementation.

Their implementation makes use of the MS cue to find a configurable number of boxes with potential objects. These boxes are then provided as input to each of the selected cue algorithms to provide their individual score for each box. The matrix of scores for each box, for each cue, is provided as input to the Bayesian framework to calculate the final score and confidence that a particular box contains an object.

During our own experimentation we found that the reliability of the boxes found is not consistent. When the algorithm is applied to the exact image twice, there is a variation in the boxes found. This is due to a sampling mechanism they implement to sample from the highest scoring boxes. In Figure 3.4 we illustrate this unreliable occurrence. This complicates the task of managing supporters and ensuring successful matches in subsequent frames. To overcome this limitation we improved the implementation as follows. Firstly, we call the function to provide many boxes of high certainty, roughly 20. We apply a K-means algorithm over these boxes to provide the top five clusters of boxes. This ensures some form of stability, especially in the scenario where many boxes are detected around a single significant object in the frame.



(a) (b)

Figure 3.4: Objectiveness sampling reliability

Secondly, we use the implementation of STC to initialise a tracker to each of the five clusters of boxes. We keep tracking these boxes for as long as they are not considered stale. To limit the number of STC trackers at any given time, we only sample for new boxes every n-th configurable iteration through the sequence. We used a basic HOG matching technique to compare boxes in subsequent frames for similarity in order to limit the possibility for drifting. If the match is within a configurable threshold, we consider it a positive tracking result. In Section 4.3 on Page 68 we show some results for using this implementations as a default real-world supporter model.

Chapter 4

Implementation and Results

4.1 Overview

In this chapter we share experimental results using the supporter model proposed in Chapter 3. We begin with experimentation on the basic supporter model using low-level generic supporters, and then evaluate the use of a more hybrid approach, taking advantage of real-world supporters, overcoming some of the shortcomings of using only low-level generic supporters.

It is important to note throughout the evaluation in this chapter, that performance was never one of the key metrics optimised, nor the evaluation of specific trackers, but rather showing the benefits of integrating context into the tracking problem.

4.2 Low-level supporters

4.2.1 Overview

In this section we focus on the most basic supporter model using only low-level generic supporters. Experiments consist of the most common and difficult challenges faced in unconstrained environments, like camera disruption, appearance changes and full occlusion.

A total of seven datasets were used with varying degrees of complexity one would expect from an unconstrained environment. We used in the basic model only SIFT supporters and the same system parameters across the different datasets. The purpose of these experiments is not to evaluate any individual tracker, but purely to illustrate the strength of our model.

For each experiment we present two figures. Firstly a sequence of images is displayed showing the model in action across a few key time intervals. The legend in Figure 4.1 can be used to interpret the image sequences.

Key	Description
	First-level supporters voting
	Second-level supporters voting
	Main tracker estimation
	Supporter model estimation

Table 4.1: Low-level supporter legend

Secondly, to illustrate the effectiveness of the model we calculate the Euclidean distance error in each frame between the ground truth and that of the model. In the event of full occlusion, if the dataset has no location data to offer, ground truth is obtained by taking a human estimate of where the target should be. We highlight the error when the tracker is operating in isolation as well as with the assistance of our model.

Table 4.2 shows the summary of experiments and results for this section.

4.2.2 Experiments

Red bull: In this sequence the camera is fixed, a person is holding a Red bull and slowly moves it across the screen. The Red bull at some point becomes completely occluded behind a large object and changes direction while occluded, not following its original path across the screen. An ISM object detector was used as the main tracker up until occlusion occurred. The model was capable of successfully tracking the target until it became visible again and the ISM object detector could resume its task.

Experiment	Short Description	Results Summary
Red bull	Target fully occluded and changes direction while occluded. Dataset from [20].	Model successfully tracked through occlusion.
Walking person	Multiple partial and full occlusions over a long sequence. Target changes direction while model is tracking. Dataset from [21].	Successful tracking under partial and full occlusion when target was walking in one direction. When target direction changed our model failed to track accurately.
Pedestrian	Target leaves the field of view resulting in full occlusion and becomes visible again after a while. Dataset from [22].	Successfully tracked the target when it left the field of vision.
FaceChange	Close up face sequence where face gets partially occluded, changes appearance and undergo multiple rotations and tilting. Dataset from [23].	Successful tracking through most appearance challenges, however tilting of head presented some inaccuracies in measurement.
Gymnast	Person detector trying to track a gymnast sequence where gymnast changes appearance continuously. Dataset from [24].	Successful tracking through most appearance changes, however sudden movements presented inaccuracies in our models measurement.
Izandri	Similar to the Red bull experiment, target fully occluded and changes direction during occlusion and becomes visible again.	Successful tracking during appearance challenges, and full occlusion. Reduced accuracy when target changes direction during occlusion and surfaces again.
Pillars	Close up view of target getting fully occluded behind pillars whiles walking. Dataset from [21].	Our model is unable to accurately track the target through occlusion.

Table 4.2: Low-level supporter summary

In Figure 4.2 when the ISM object detector no longer detects the target we set the target position coordinate to (0,0). Although it shows that our model reduced the error, the graph does not capture the real improvement of tracking through occlusion as you would see from the sequence of images in Figure 4.1.

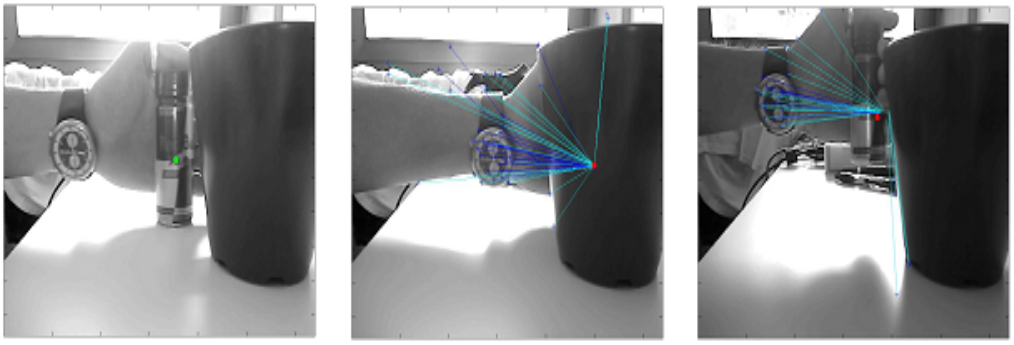


Figure 4.1: Red bull sequence

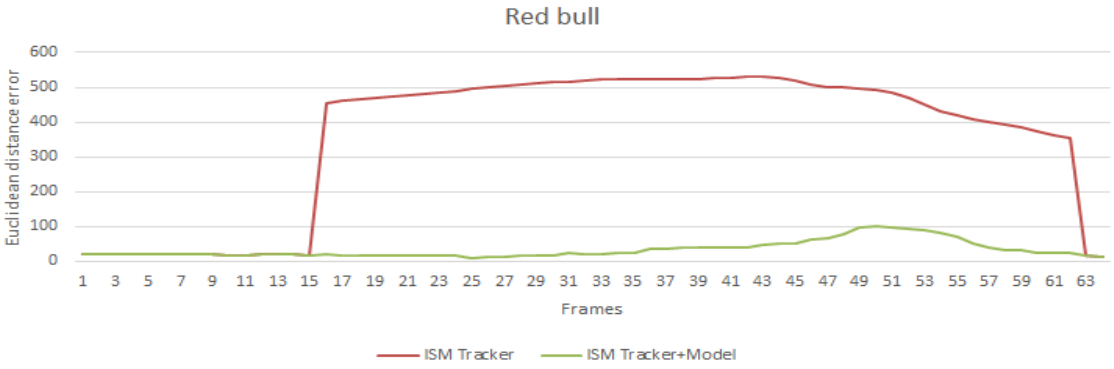
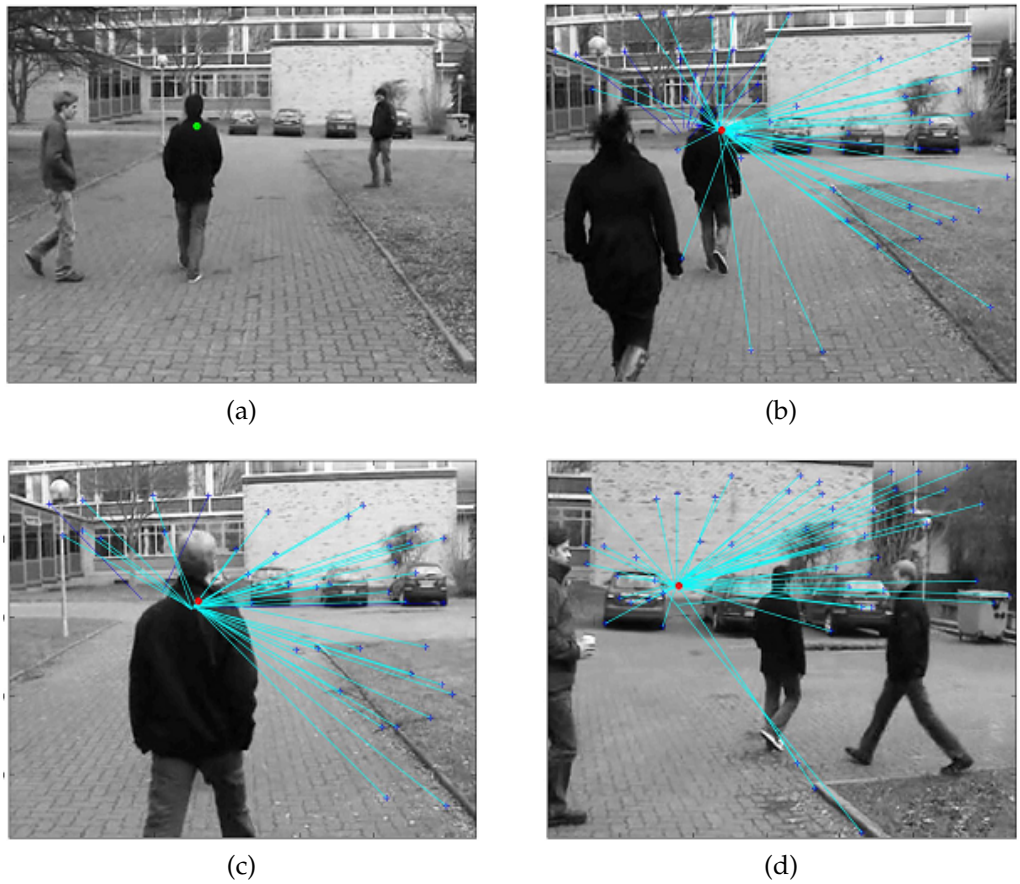


Figure 4.2: Red bull Euclidean distance error

Walking person: We deal in this sequence with smooth camera movement while following a target individual. The target walks along a path with random people crossing between the camera and target resulting in short periods of occlusion. We used a particle filter as the main tracker. Figure 4.3c shows how the model was still capable of tracking the target while being occluded but in Figure 4.3d when the target changes direction while being driven by our model, it was no longer able to stay with the target.

When the target was turning right and walking across the screen, there were not enough correlated low-level supporters that moves with the target that can be used during voting. The majority of good supporters are static objects part of the background. In Section 4.3 we show that real-world supporters provided some improvement. The people that do happen to walk during that time frame were either walking in the opposite direction, or the same direction but at different velocity.

In Figure 4.4, the green arrow on the Euclidean distance error graph represents roughly the time when multiple occlusion started to occur more frequently from random people crossing in front of the camera. Our model was still capable of staying relative on the target until the point represented by the red arrow. The red arrow indicates where the target started turning right, walking horizontally across the screen. At that stage with or without our model, the estimated position drifted from the target. The distance error for the particle filter only, might look that it is performing better than without our model, but was in fact drifting and just happen to do so closer to the target. The position however cannot be trusted.



(a) (b)

(c) (d)

Figure 4.3: Walking person sequence

Pedestrian: A camera focuses on a group of pedestrians walking in a parking lot with the target pedestrian a few meters behind three other pedestrians. The camera is not fixed and undergoes a lot of movement. At one

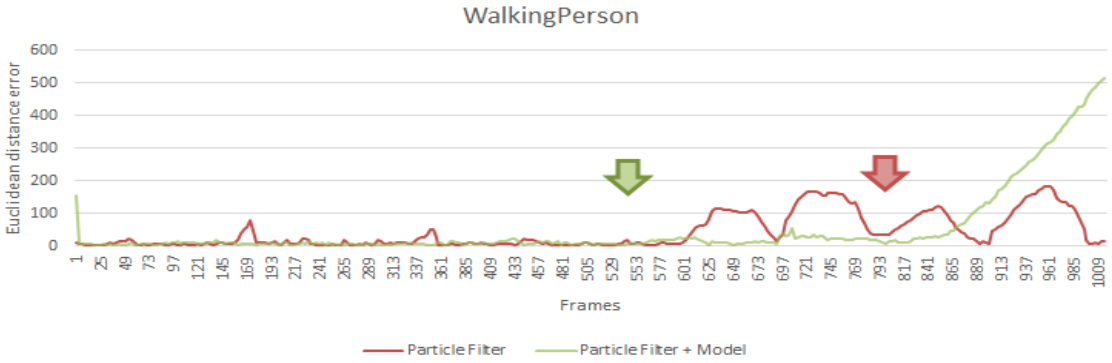
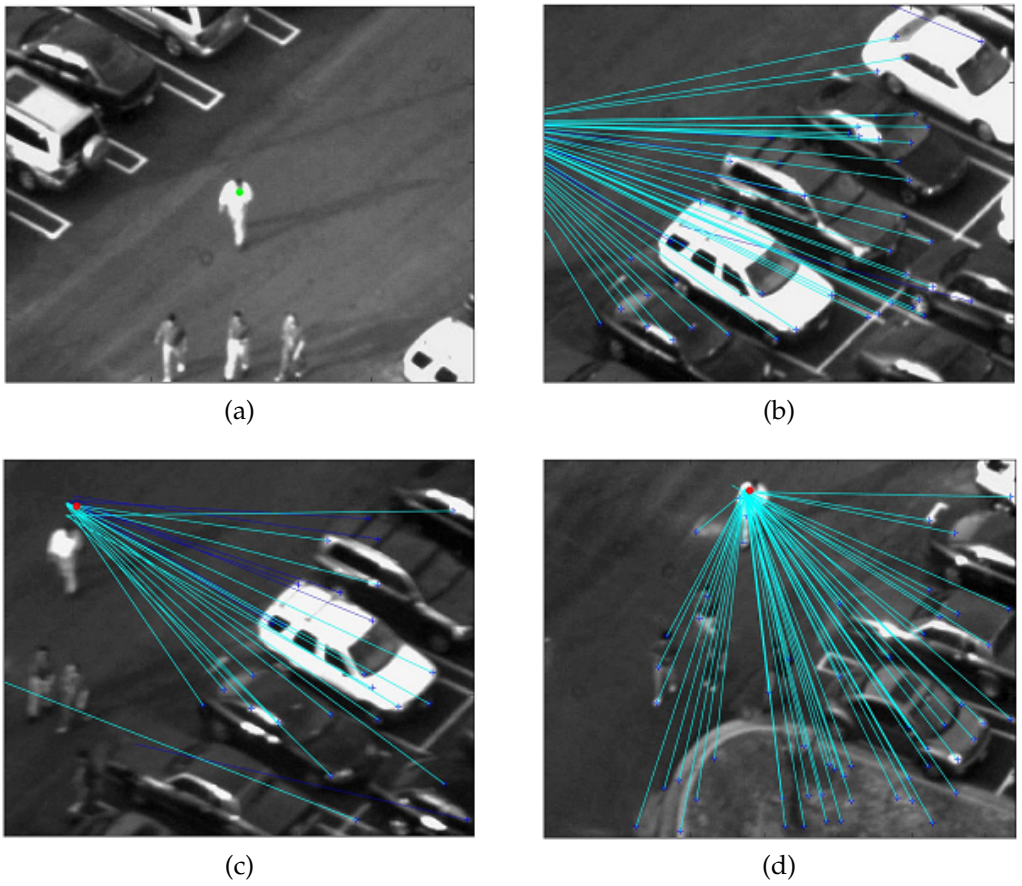


Figure 4.4: Walking person Euclidean distance error

stage the camera moves so far to the right that the target leaves the field of view (FOV). A particle filter was used to track the target to the point where it leaves the screen, the supporter model successfully keeps tracking the target outside the field of view until it reappears. The particle filter does not continue tracking when the target becomes visible and our model continues tracking. As some point our model was off by a small error, see Figure 4.5c, but still manages to stay close to the target position. It eventually finds the target again, see Figure 4.5d. The Euclidean distance error graph is shown in Figure 4.7.

To illustrate the model's verification properties we disabled the verification function. In Figure 4.6a the model was tracking the target successfully when out of view, but the particle filter picked up on a similar colour than the target and incorrectly jumped to the motor vehicle, Figure 4.6b. With the verification function enabled a jump of this magnitude will not be allowed and our model will be used instead. It is important to remember that the verification function uses relative distances making it extremely robust to camera movement and abrupt supporter movements.

FaceChange: The sequence consists of a fixed camera and a person's face changing appearance by means of rotation, tilting and partial occlusion from external objects. We used an off-the-shelf face detector in Matlab based on Viola and Jones [25] as our main tracker/detector. In the event of too much change in the target's appearance, the supporter model took over and managed to keep track relatively successfully, except for the instances where



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

Figure 4.5: Pedestrian sequence

the face tilted left to right with the rest of his body remaining static. See Figure 4.8c. The most confident supporters on the person's face itself were overridden by the other many supporters external to the face and part of the background. In Section 4.3 we focus on the specific subset of frames where the face is tilting and show how using real-world, more predictable supporters improve the outcome.

In Figure 4.9, every time the face detector was unable to detect the target it is illustrated by a spike. A miss essentially resulted in a target coordinate of (0,0) creating spikes in the graph.

In this experiment we show again the verification properties of the supporter model. The face detector on few occasions makes a false detection and finds a second face in the image. See Figure 4.10. The supporter model votes at the start of each iteration and combining that with the face detec-

(a) (b)

Figure 4.6: Pedestrian without verification

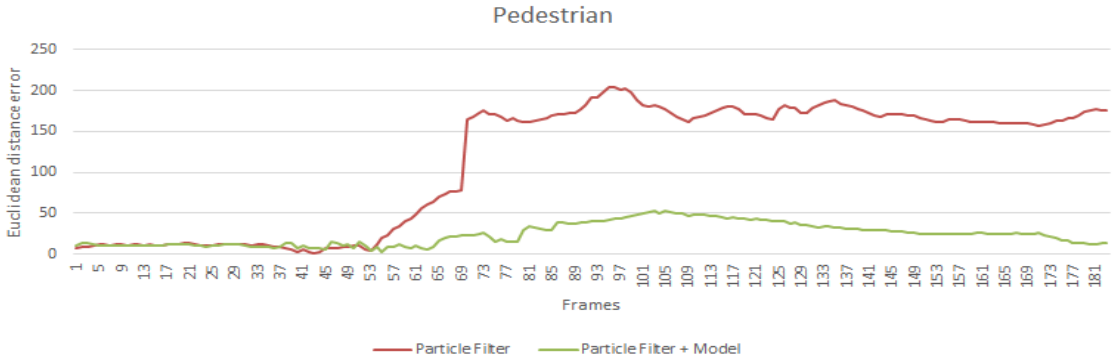
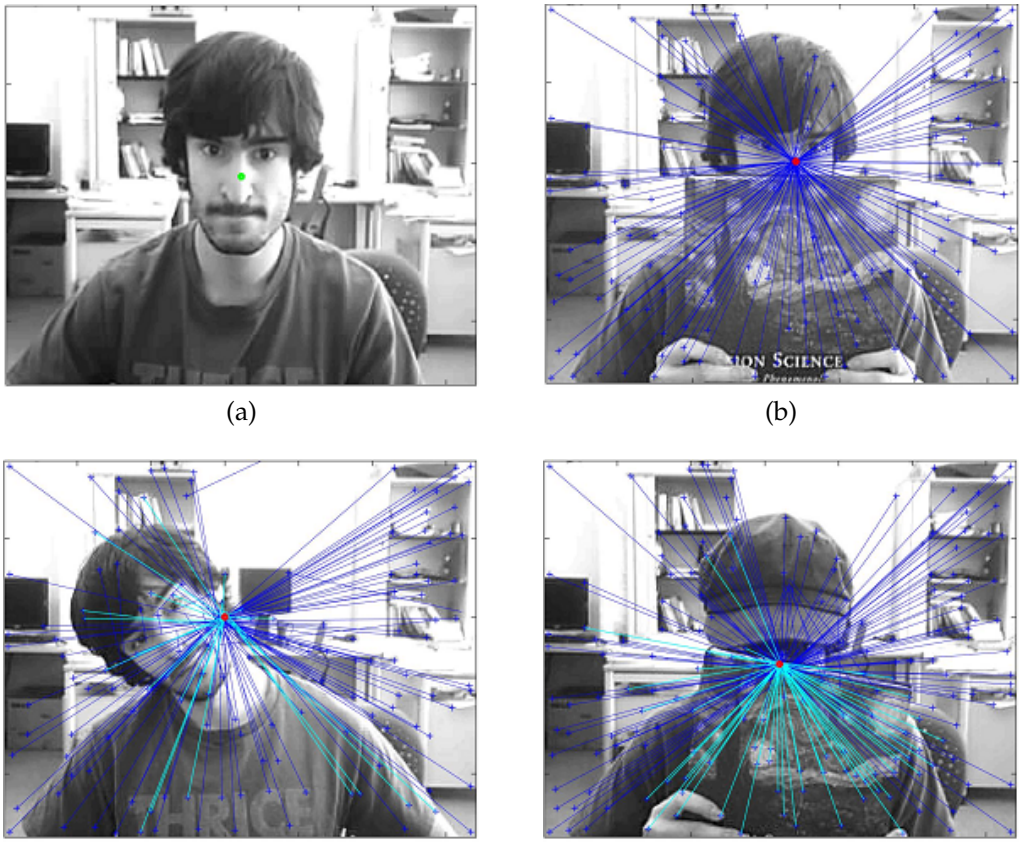


Figure 4.7: Pedestrian Euclidean distance error

tor output, it helps distinguish which of the two boxes are most likely the target. Without the supporter model to verify, the typical solution around this scenario would be to initialise some form of tracker along with the face detector to help in the event of duplicate objects. The use of a tracker however limits the robustness as mentioned in Section 3.4.2, especially with a moving camera.

Gymnast: In this experiment the use of a particle filter or STC tracker resulted in too much drifting. The STC tracker refers to the work done in Zhang *et al.* [14]. If the target tracker drifts, our model will learn this false positive and will provide no value in the event it needs to estimate the target position. Instead we used an off-the-shelf person detector in Matlab to detect a gymnast performing a floor sequence. The person detector in Matlab is based on the work done in Dalal and Triggs [26]. As stated in Section



(a) (b)

(c) (d)

Figure 4.8: FaceChange sequence

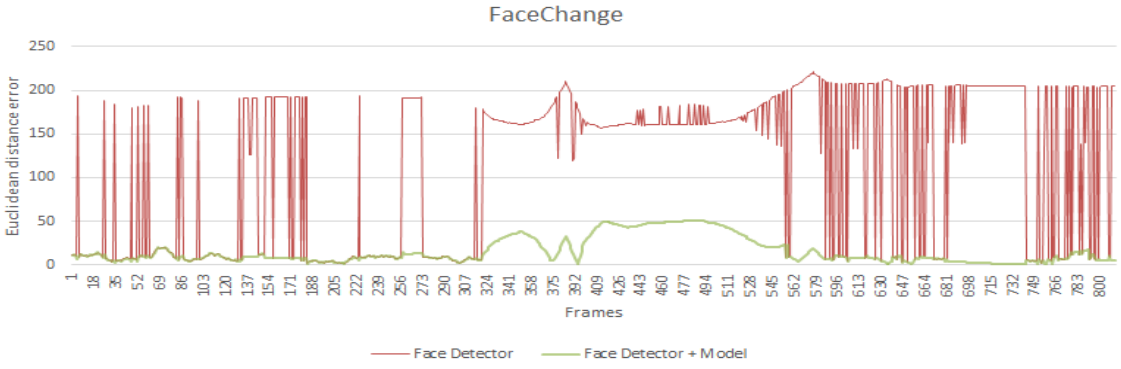


Figure 4.9: FaceChange Euclidean distance error

3.2 we want a context flexible model. Here we show again how flexible our model is in integrating any existing tracker/detector into the existing tracking solution.

The target undergoes multiple aggressive appearance changes, resulting in

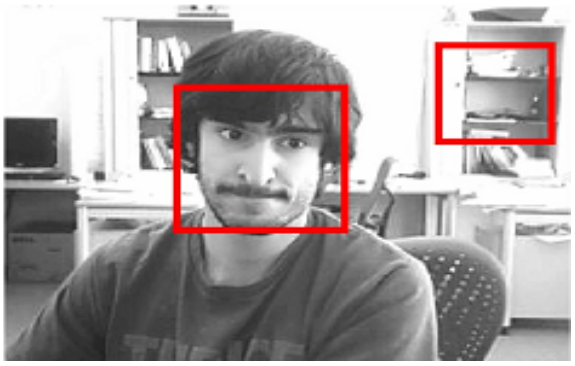


Figure 4.10: FaceChange without verification

no detection. In these events the supporter model stayed relatively easy on the target with the odd instance where proper correlated context was not sufficient, see Figure 4.11c. In this particular sequence the use of real-world supporters would not have added much value, due to the fact that the only relevant real-world object in the scene that has any movement is the gymnast. The large number of appearance changes also leaves little room for finding real-world objects on the gymnast. All other supporters are static to the gymnast's movement and are part of the background.

In Figure 4.12, similar to experiment ***FaceChange***, when the person detector was unable to detect the target it is illustrated by a spike. A miss essentially resulted in a target coordinate of (0,0) creating spikes in the graph.

Izandri: In this sequence the target remains still and instead the camera changes its position. A particle filter was used for the main target tracker and very early on in the sequence the particle filter loses track of the target intermittently. During these brief appearance challenges (light exposure) our model was capable of tracking the target successfully. At some point the camera moves in and out behind a counter losing complete visibility of the target. As the camera drops behind the counter our model takes over and stays with the target. See Figure 4.13c. When the camera finally changes direction moving out from behind the counter it remains with the target but the error is rather big. See Figure 4.14. The model is supposed to be tracking where the green arrow is pointing.

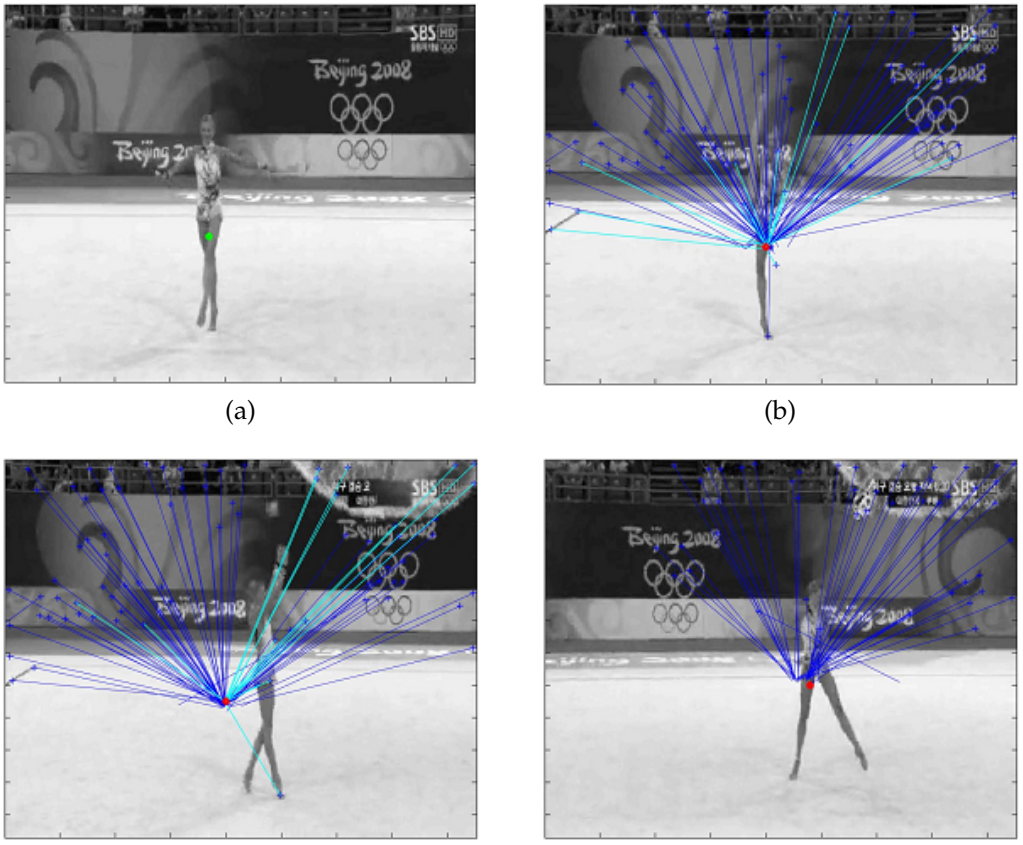


Figure 4.11: Gymnast sequence

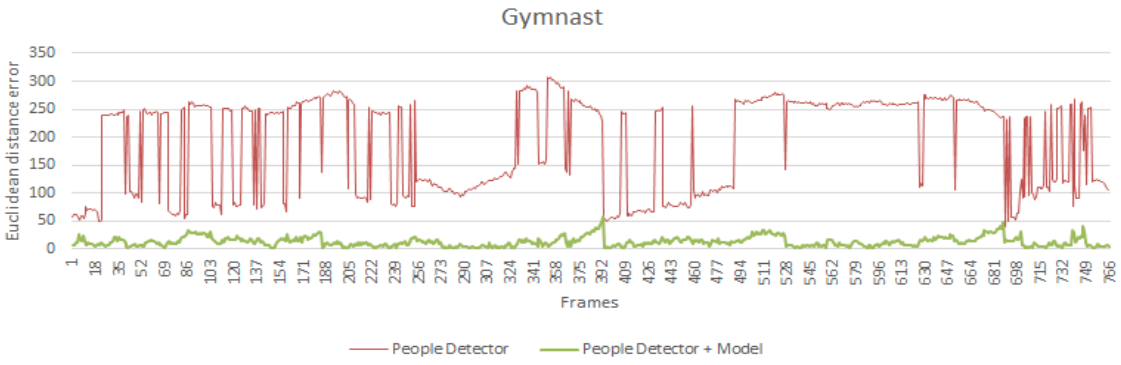
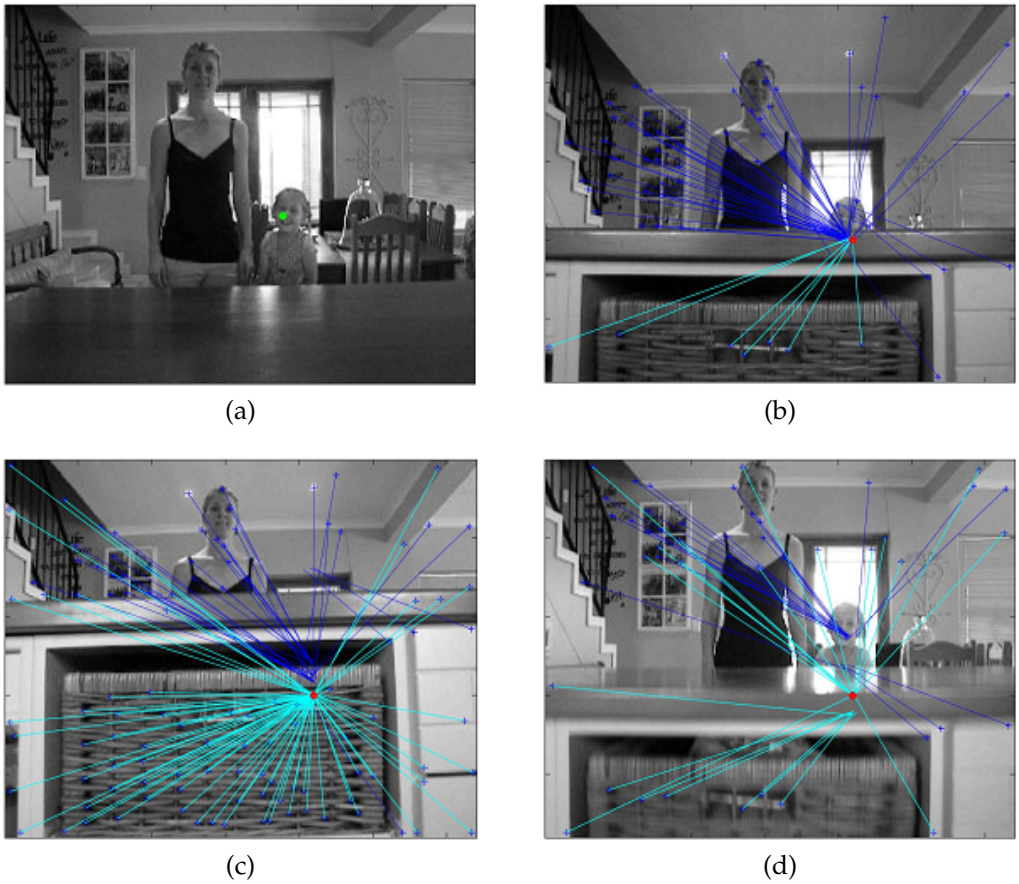


Figure 4.12: Gymnast Euclidean distance error

The reason for this error is due to the large number of second-level supporters that was trained during occlusion. The second-level supporters out weighted the voting of the few remaining first-level supporters, especially when the camera was taking time to change direction. The second-level

supporters trained during that window of time when movement was static pointed to the target correctly. When the camera finally moves again those many static background supporters containing extremely good correlation with the target are now used to vote as the camera moves. The number of good correlated second-level supporters outweighed even the impact of our priority filter. The Euclidean distance error graph is shown in Figure 4.15.

This is an example where quality is more valuable than quantity. Using only a few real-world objects would be more valuable. Too many low-level supporters causes noise, especially if they are static and part of the background. In Section 4.3 experiment ***Izandri2*** we show how this problem is resolved using real-world supporters.



(a) (b)

(c) (d)

Figure 4.13: Izandri sequence

Pillars: In this experiment the target is walking in a straight line while the camera follows him at the same velocity at a fixed distance. The target walks

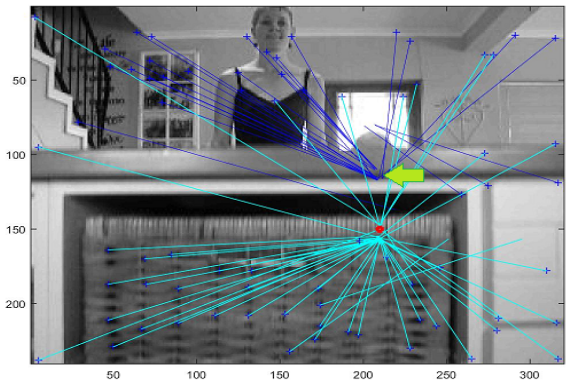


Figure 4.14: Izandri low-level supporter challenges

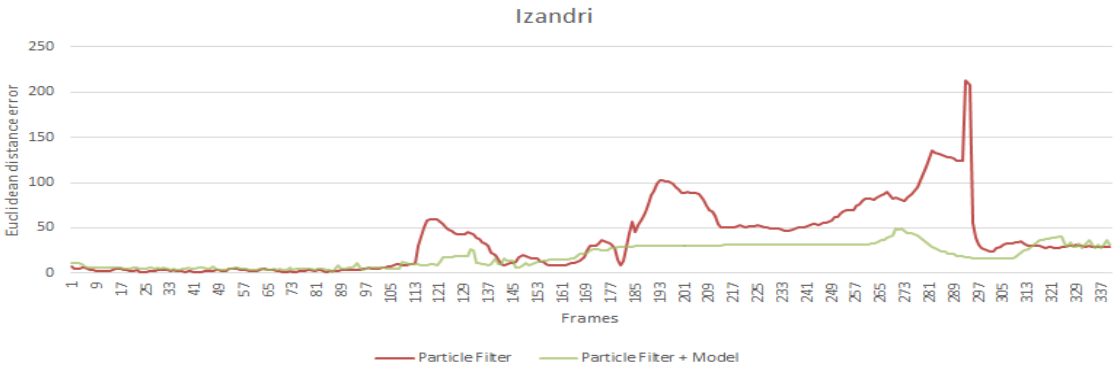
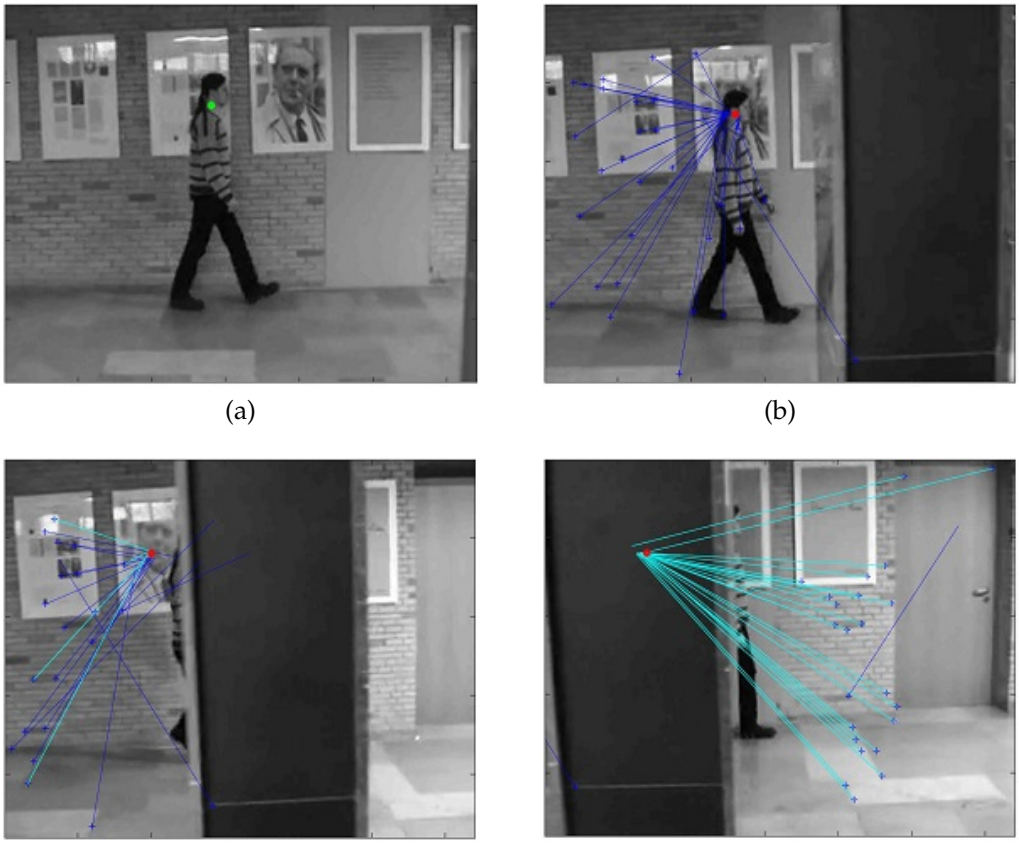


Figure 4.15: Izandri Euclidean distance error

behind a pillar getting fully occluded. A particle filter was used to drive the tracker and unfortunately our model was not capable of tracking successfully through the occlusion around Frame 143. See Figure 4.16c. Firstly the pillar causing the occlusion takes up too much of the image not leaving enough room for proper supporters to be used. Secondly the majority of supporters available are static and part of the background with low correlation to the target. When the target became visible again, neither the tracker nor our model caught up with it again. The use of the real-world Objectness model could not improve the results. The Euclidean distance error graph is shown in Figure 4.17.



(a) (b)

(c) (d)

Figure 4.16: Pillars sequence

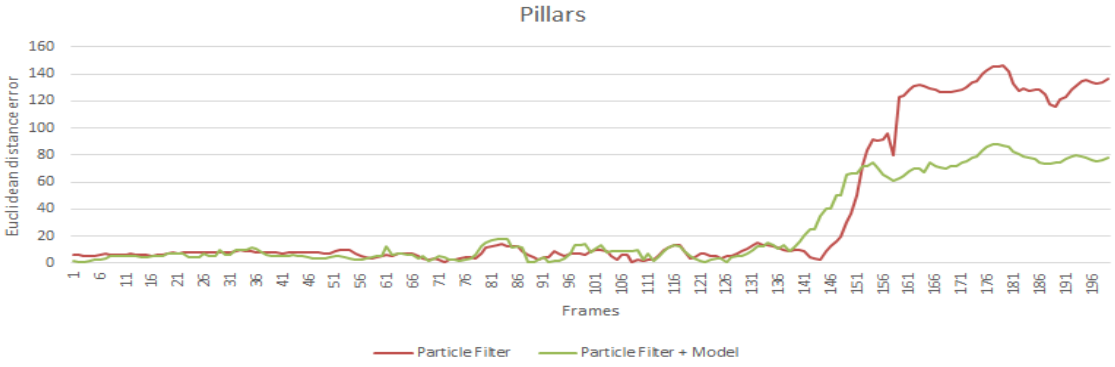


Figure 4.17: Pillars Euclidean distance error

4.3 Real-world supporters

4.3.1 Overview

Higher-level scene understanding by definition is very vague and can place one on a path of perhaps over thinking the solution by trying all kinds of

learning and understanding of how certain types of objects interact with one another. This is all valuable information that will most definitely improve the tracker, but in the end focusing only on what is real-world objects, ignoring the class of these objects, proves to be more simple and elegant and more than capable. The Objectness implementation integrated in Section 3.4.8 is a good example of such a class-less real-world object detector. Real-world objects capture object structure information and ensures for more predictable movement in comparison to low-level features like local points.

As specified in Section 3.2 we set out to simplify the use of higher-level scene understanding by only focusing on the correlated movement of real-world supporters with the target. This way we benefit from higher-level scene understanding but remain flexible with our solution, no matter how we decide to find real-world supporters.

In this section we share some experiments where real-world supporters improved the overall accuracy of the model. We also show how easily the model can be integrated with any off-the-shelf tracker/object detector.

In the experiments below the same rules apply as in Section 4.2. The same system parameters were used throughout the experiments. Each experiment is accompanied by a figure illustrating the sequence of time as well as a figure showing the Euclidean distance error graph. Figure 4.3 provides further detail on how to interpret the image sequences in this section.

Key	Description
	First-level supporters voting
	First-level real-world supporters voting
	Second-level supporters voting
	Real-world supporter detection
	Real-world supporter false detection
	Main tracker estimation
	Supporter model estimation

Table 4.3: Real-world supporter legend

Table 4.4 shows the summary of experiments and results for this section.

Experiment	Short Description	Results Summary
Walking Person2	Target changes direction while model is tracking. Dataset from [21]	The background supporters caused the model to drift, real-world supporters eventually finds target again.
Face Change2	Tilting head causing low-level static background supporters to estimate poorly. Dataset from [23]	Incorporation of real-world supports (eyes, mouth, nose) improved the target estimation.
Izandri2	Target changes direction while occluded. When target velocity reaches zero, large number of second-level supporters get learned and reduced the accuracy of model when target moves again.	Providing the real-world supporter with more predictable movement a higher weighting in the final vote improved, the tracking error.
Peter Rabbit	Appearance changes occur on active supporters while target leaves the FOV.	Real-world supporters provides more predictable behaviour during appearance changes.

Table 4.4: Real-world supporter summary

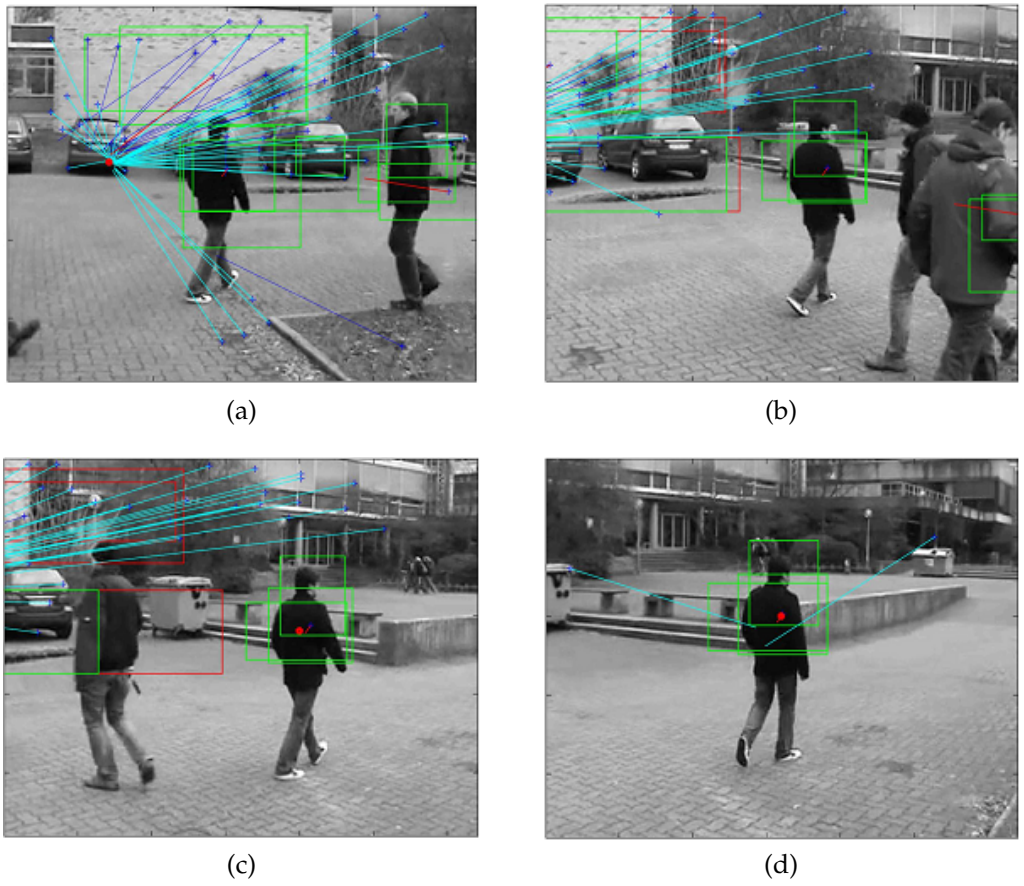
4.3.2 Experiments

Walking Person2: This particular sequence in Section 4.2.2 presented some challenges when the target changes direction while being tracked by our model. There were not enough correlated low-level supporters to use for voting; the majority of the context was part of the background. The few potential supporters that happen to move with the target, were either doing so at a different velocity or in an opposite direction.

In this experiment we focus on the subset of frames prior to when the target turns direction all the way to the end. We use the class-less object detector of Alexe *et al.* [10] and used a weighting of **40%** SIFT and **60%** Objectness.

In Figure 4.19, the red arrow represents roughly the time when the target changed direction. Both low-level and real-world supporters were unable to stay with the target due to all the background supporters. See Figure 4.18a. There are however still some trained real-world supporters that were

not available when we only trained low-level supporters. At some point, (see green arrow), the target moves far enough to the right for the background supporters to leave the frame at which point some of the remaining real-world supporters provide sufficient voting contribution to find our target again all the way to the end. See Section 4.18c. In some repetitions of this experiment the real-world supporters discovered using Objectness were capable of staying with the target through the entire sequence.



(a) (b)

(c) (d)

Figure 4.18: Walking person real-world sequence

FaceChange2: In Section 4.2.2 we indicated in the ***FaceChange*** experiment that there was a particular subset in the sequence where our model did not fare that well due to the person's head tilting to the sides. The number of background supporters overwhelms the few good supporters on the target's face. In this experiment we focus only on that particular section in time and how the incorporation of real-world supporters can improve the

Figure 4.19: Walking Person Euclidean distance error comparison

accuracy. We manually selected three distinct features, the left eye, right eye and the nose and mouth together. We used the STC tracker to manage the tracking/detection of these supporters. The process of manual selection can easily be replaced using for example the object detector in Viola and Jones [25] to find these distinct features and then initialising an STC tracker to track. We used a weighting of **40%** SIFT and **60%** STC tracker in this experiment, to give the facial features just a slight advantage over the low-level supporters. In this sequence the use of the real-world Objectness model was too generic and did not provide any improvement. Using a class-specific supporter model focusing on the eyes and specific facial features was more valuable.

Figure 4.20a and 4.20b represents two distinct failures when using SIFT only, in Figure 4.20c and 4.20d we show the same frames but with the use of real-world supporters. In Figure 4.20c the detection of the two eyes and the combination of mouth and nose are indicated using green boxes. It is clear looking at Figure 4.20a and 4.20c that the error is reduced and is more realistic/trustworthy. In Figure 4.20d the tracking of the eyes drifted to a point where they were no longer detected, therefore the red boxes. Only the mouth and nose combination was detected in that frame (green box) and with the assistance of second-level supporters provided a more accurate position of the target than in Figure 4.20b.

In Figure 4.21 we show the reduction in the error when incorporating real-world supporters.

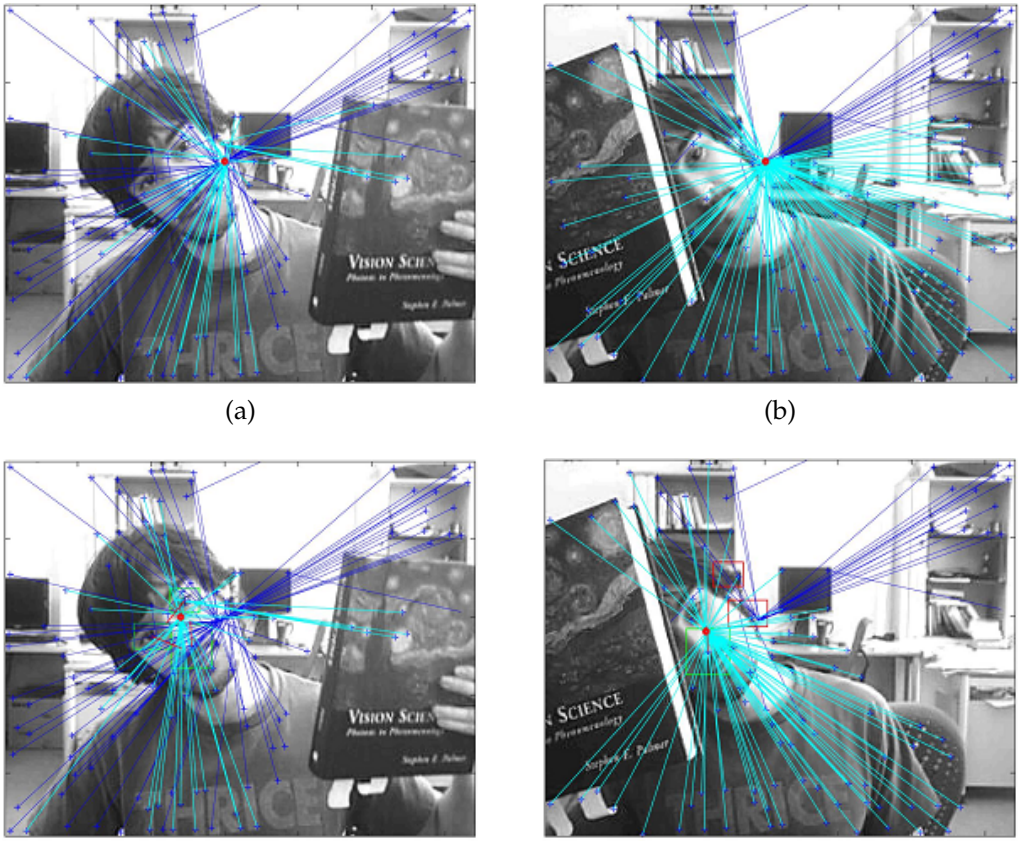


Figure 4.20: FaceChange real-world sequence

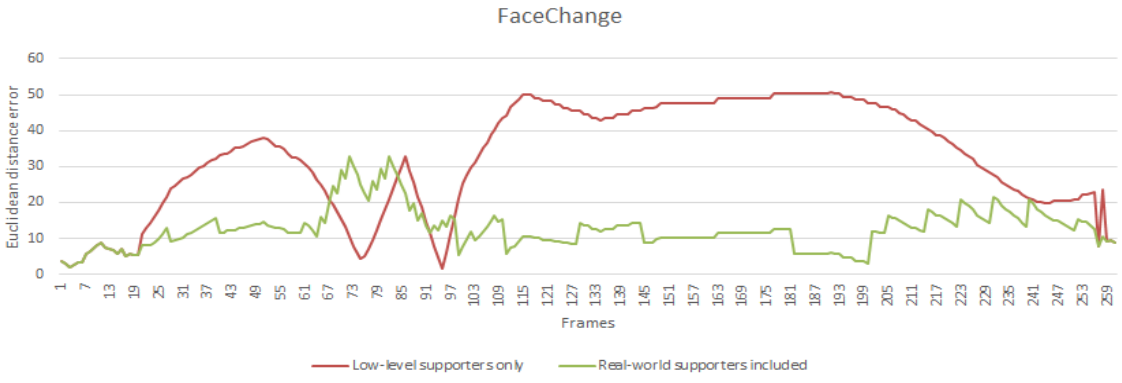


Figure 4.21: FaceChange Euclidean distance error comparison

Izandri2: In Section 4.2.2, experiment ***Izandri***, we showed the problem presented with training too many second-level supporters during a window when the camera changes direction. During that brief time when the camera changes direction, it reduces its velocity to zero. At that stage a large num-

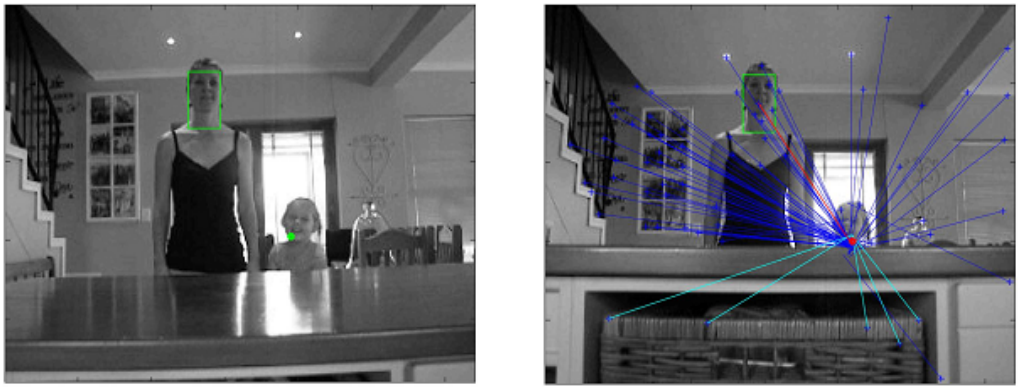
ber of second-level supporters is trained. The longer the pause in movement the stronger these supporters become. When the camera moves again, the overwhelming number of second-level supporters outweighs the few remaining first-level supporters. Similar challenge was found in the ***WalkingPerson*** experiment, fortunately in this experiment there are real-world supporters that we can use to turn the focus to quality supporters rather than quantity.

In this experiment we used the STC tracker to track the real-world supporter, the mother's face. A weighting of **40%** SIFT and **60%** STC was used. This decision was made so that the SIFT model still plays an integral role but to give the high level STC tracker a slight advantage. In Figure 4.23 and 4.22d it is clear that the mother's face plays a role in estimating more accurately where the child's face might be. This reduction in error is also visible from the Euclidean distance error in Figure 4.24.

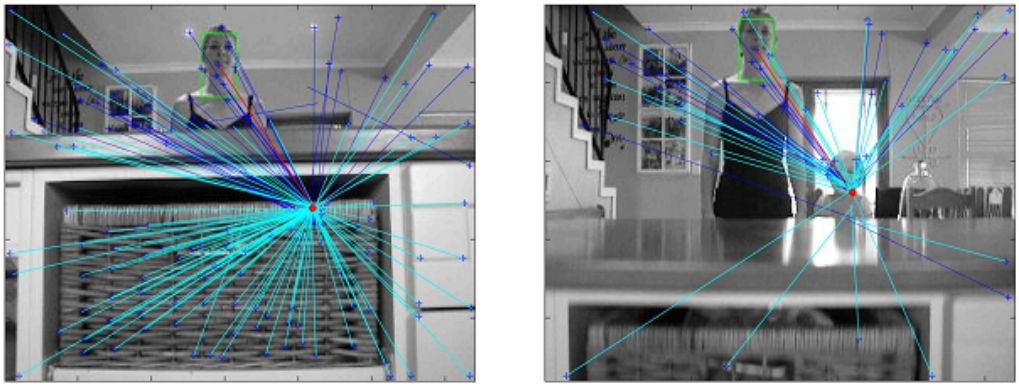
We also tested the sequence with our default real-world Objectness supporter model and it performed better than with low-level supporters only. Our estimation however did start to drift somewhat closer to the end of the sequence due to the STC tracker on one of the key Objectness boxes drifting slightly. As stated in Section 3.4.8, if the Objectness implementation improves to be more reliable in detecting potential objects in subsequent frames, the tracking of potential boxes can be replaced by detection only, reducing the chances of drifting.

Peter Rabbit: The following experiment is a basic sequence where we try to illustrate the importance of having real-world supporters, especially in the face of aggressive appearance change. The target (Peter Rabbit) is tracked using the STC tracker. We force the main target tracker to stop tracking so that we can illustrate our point.

In the first iteration, (see Figure 4.25), we only make use of low-level supporters like SIFT. Once the main tracker is disabled, our model steps in and stays with the target despite moving the camera position, forcing the target to leave the field of view. The majority of supporters contributing to the



(a) (b)



(c) (d)

Figure 4.22: Izandri real-world sequence

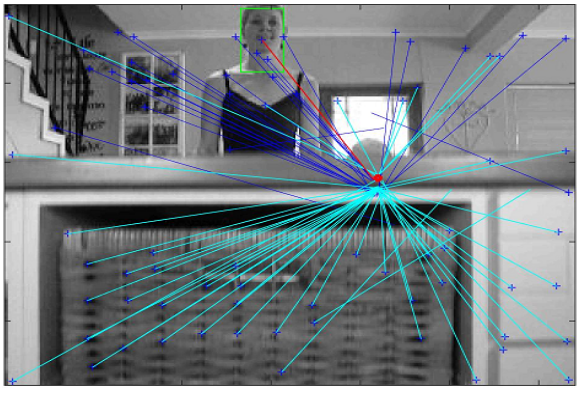


Figure 4.23: Izandri real-world supporter improvement

voting space is from a teddy bear on the far right. When we suddenly turn the teddy around 180 degrees, the majority of SIFT supporters disappears and our model loses track of the target.

For the second iteration we focus on the teddy as a real-world supporter

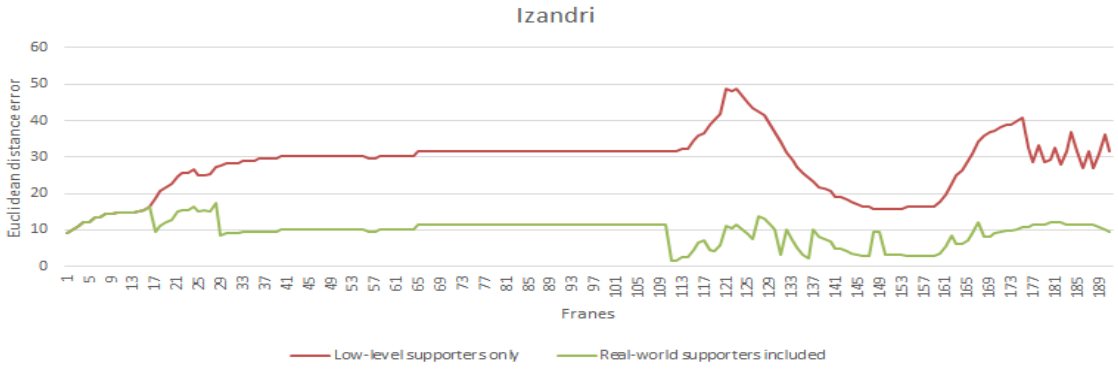
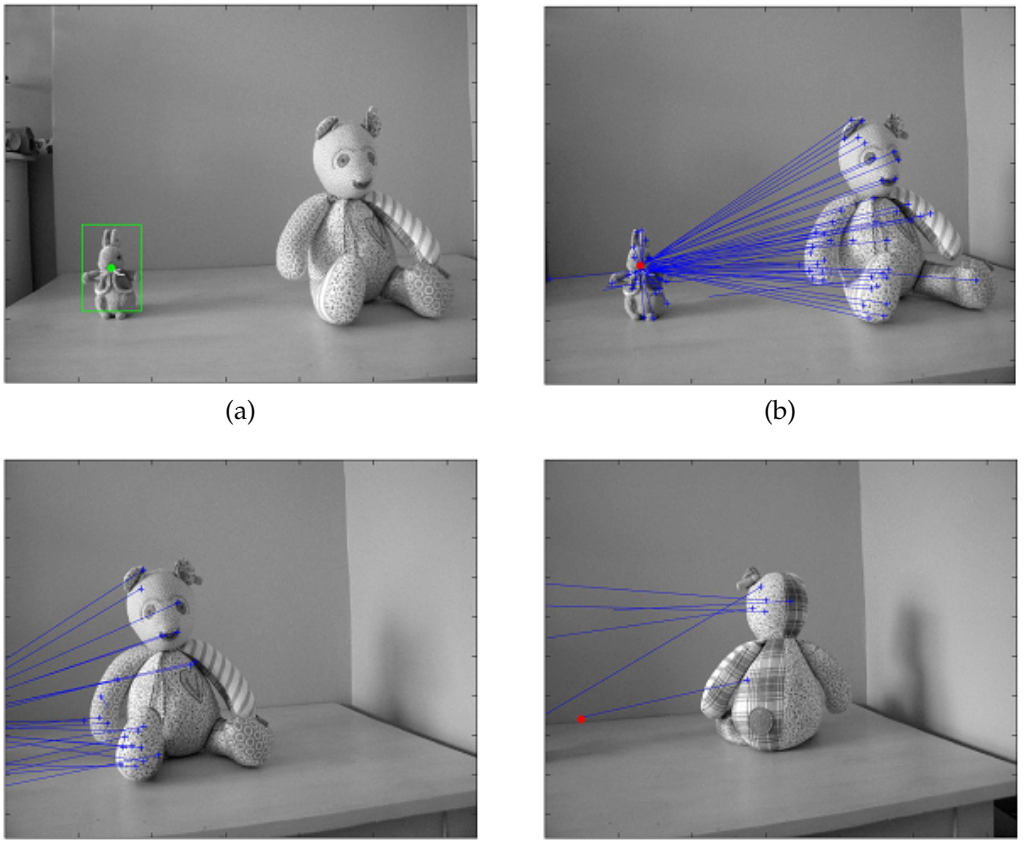


Figure 4.24: Izandri Euclidean distance error comparison

and set the weighting to **40%** SIFT and **60%** Objectness. Similar to the first iteration, our model stays with the target even when leaving the field of view. However this time when the teddy turns 180 degrees, the Objectness supporter model tracking potential objects like the teddy, provides a more reliable vote from a real-world source of context. In Figure 4.27 we show the Euclidean distance error graph for the first and second iteration. Even though the error differences between both sequences is not that high, the supporters contributing to the voting space in Figure 4.26d shows a more reliable source. In the low-level sequence (Figure 4.25d), the matched SIFT supporter that provides the final vote, is a mismatch. They just happen to contribute a relative close estimation to the real target position. In the real-world sequence, matching the Objectness boxes is a positive match and can be trusted.

4.4 Second-level supporters

In this section we conduct three experiments to demonstrate the usefulness of second-level supporters. We use the same system parameters as in previous experiments. To test the effectiveness of second-level supporters we disable the main target tracker after 5 frames. This is to ensure that we at least have some first-level supporters that will help train second-level supporters and then see how long this self-learning model can carry on tracking the target.



(a) (b)

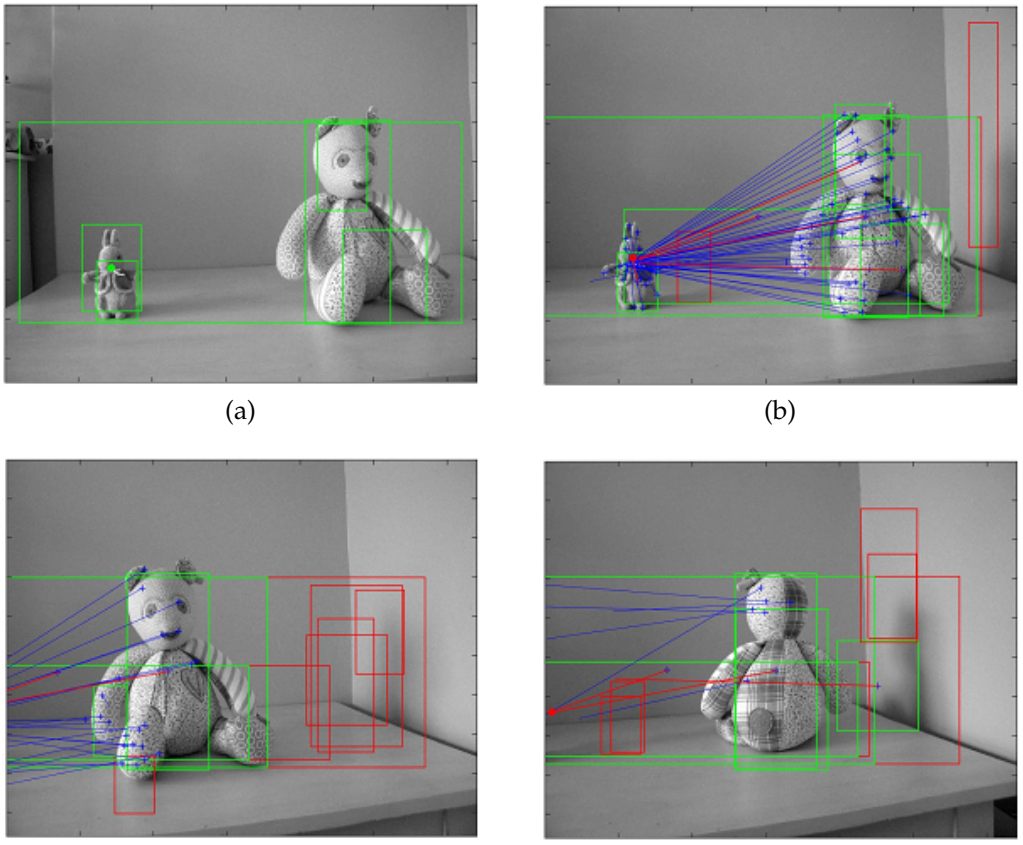
(c) (d)

Figure 4.25: Peter Rabbit low-level sequence

The blue arrows in the Euclidean distance error graphs below indicate where the first-level supporters became extinct and the second-level supporters took over the tracking task.

Jogging: In this sequence (dataset from [27]), two joggers are running on a road, see Figure 4.29. The target gets occluded for a short period as it passes behind a street light. The second-level supporters were capable of staying on the target all the way through the occlusion but started drifting a few frames after. There were too few correlated supporters to be effective and too many static background supporters. The Euclidean distance error graph is shown in Figure 4.29.

Football1: In this sequence (dataset from [28]) a footballer runs with the ball through a crowd of opponents and his own team, see Figure 4.30. The footballer undergoes multiple appearance changes as well as partial occlusions.



(a) (b)

(c) (d)

Figure 4.26: Peter Rabbit real-world sequence

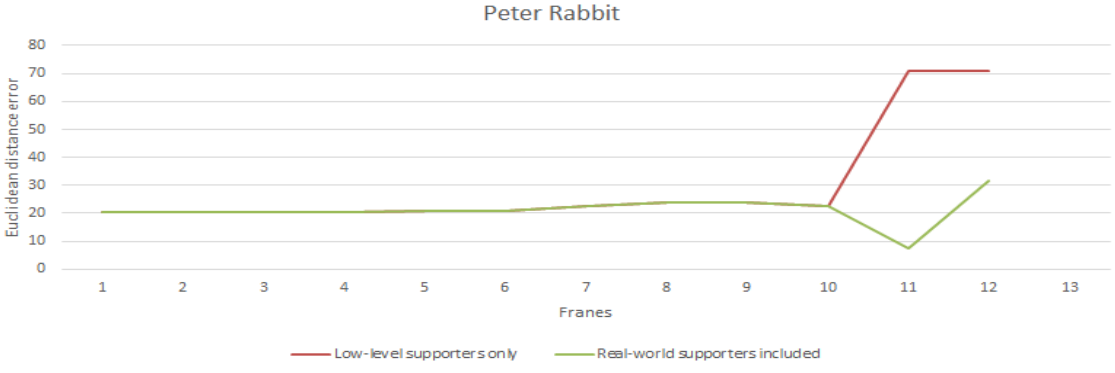


Figure 4.27: Peter Rabbit Euclidean distance error comparison

The second-level supporters that gets trained the moment the main target model gets disabled were capable of staying relatively close to the target for the entire sequence. The priority filter described in Section 3.4.6 on Page 49 played a key role here.

Figure 4.28: Jogging sequence

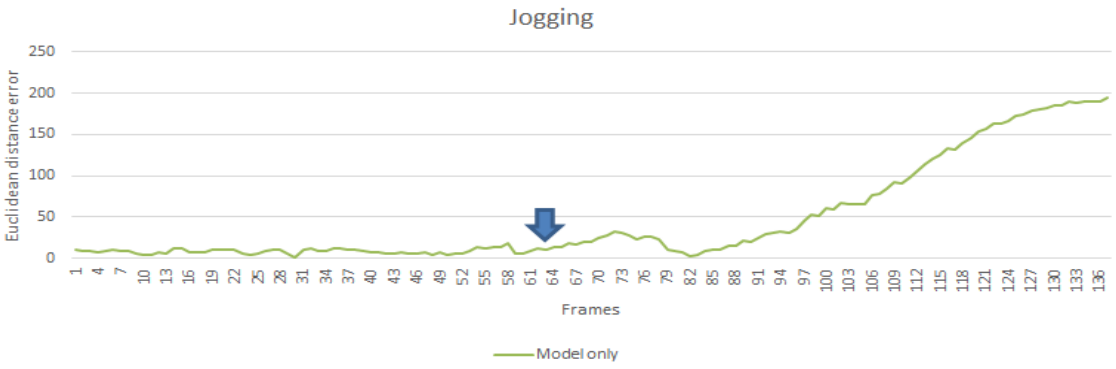


Figure 4.29: Jogging Euclidean distance error

In Figure 4.31 the success of the second-level supporters is not properly visible. When the second-level supporters took over the target estimation slightly drifted from the target's head to his chest, illustrated by the sudden increase in error. However, tracking remained on the target's chest and stayed relatively on the target for the entire sequence as illustrated.

Football2: A footballer runs backwards in this sequence (dataset from [29]), and gets partially occluded by other players as well as undergoes multiple appearance changes. The second-level supporters were able to stay on target for a while but eventually drift.

The yellow arrow in Figure 4.32 shows the importance of correlated movement. Even though we still had plenty of first-level supporters available, the second-level supporters were of more value due to their correlation to the target. It is important to note that the priority filter we implemented

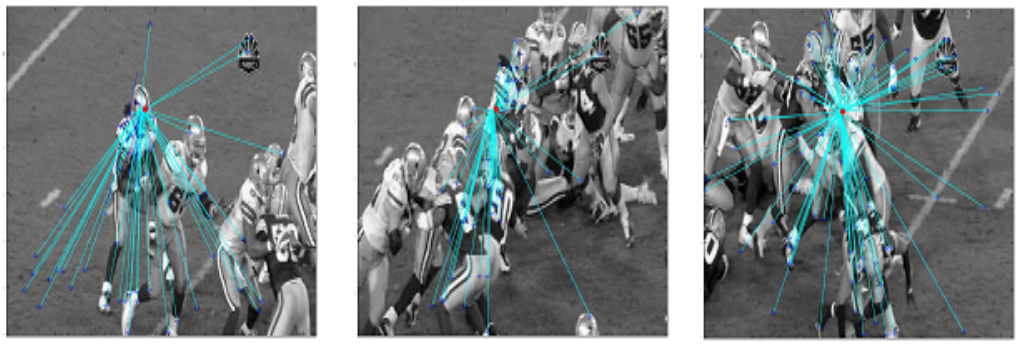


Figure 4.30: Football1 sequence

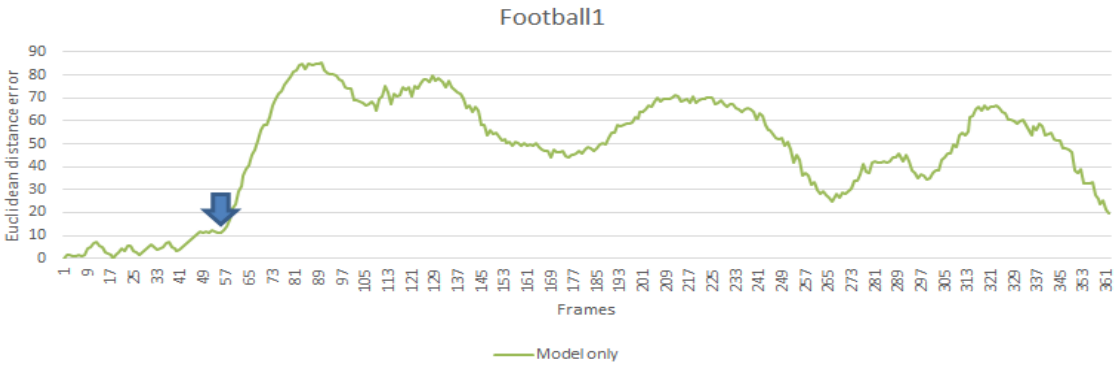


Figure 4.31: Football1 Euclidean distance error

(see Section 3.4.6 on Page 49), does not negate the voting done by correlated second-level supporters over uncorrelated first-level supporters. The Euclidean distance error graph is shown in Figure 4.33.

Figure 4.32: Football2 sequence

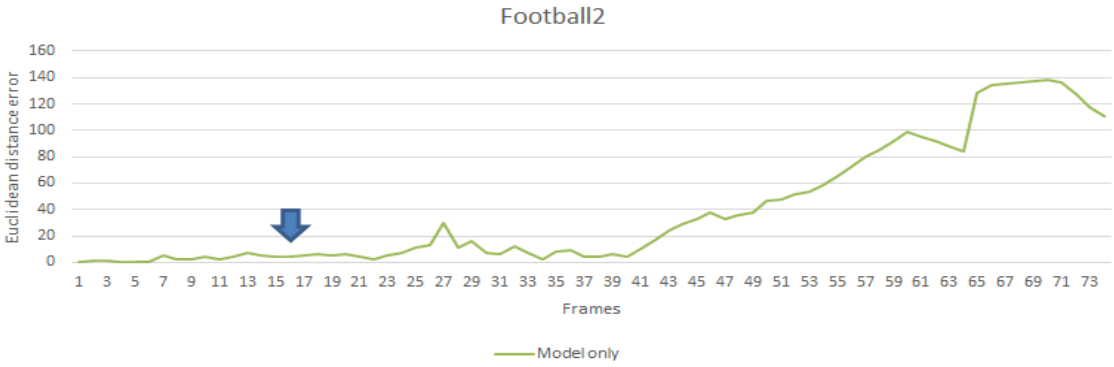


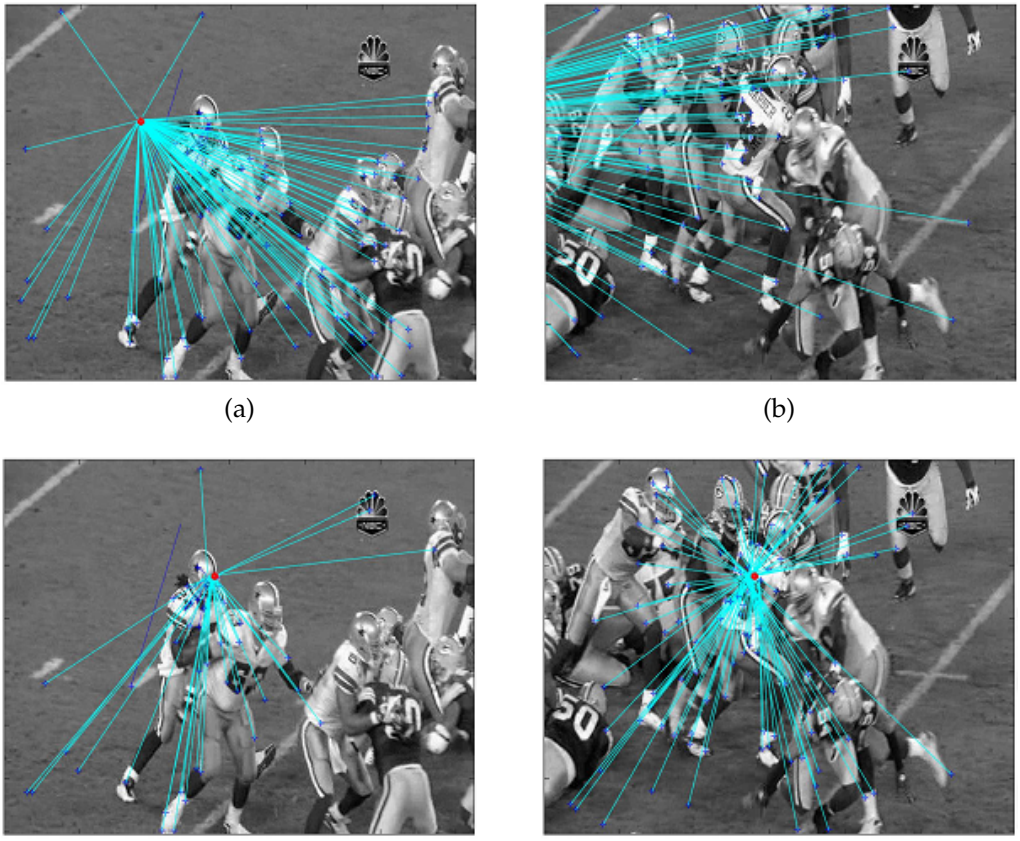
Figure 4.33: Football2 Euclidean distance error

4.5 Priority filter

A large numbers of second-level supporters can potentially cause drifting in the presence of good trustworthy first-level supporters. As mentioned in Section 3.4.7 this was discovered during experimentation on the strength of second-level supporters. In this section we compare two sequences, ***Football1*** and ***Football2***, with and without our priority filter.

Figures 4.34a and 4.34b represent the sequence without the use of a filter. It is clear from Figure 4.34a that there are still good first-level supporters on the helmet that were ignored. In Figure 4.34c the use of our filter allowed for the tracker to remain on target a while longer ensuring we train proper second-level supporters for as long as possible with the guidance of good first-level supporters. The final result is clear when comparing 4.34b and 4.34d.

Figures 4.35 and 4.36 illustrate the Euclidean distance error comparison with and without the use of a filter for the two football sequences. In Figure 4.36, the error near the end of the sequence seems to decrease/improve without the use of a filter. At that time the tracker was blindly drifting around and could not be trusted.



(a) (b)

(c) (d)

Figure 4.34: Football1 filter comparison sequence

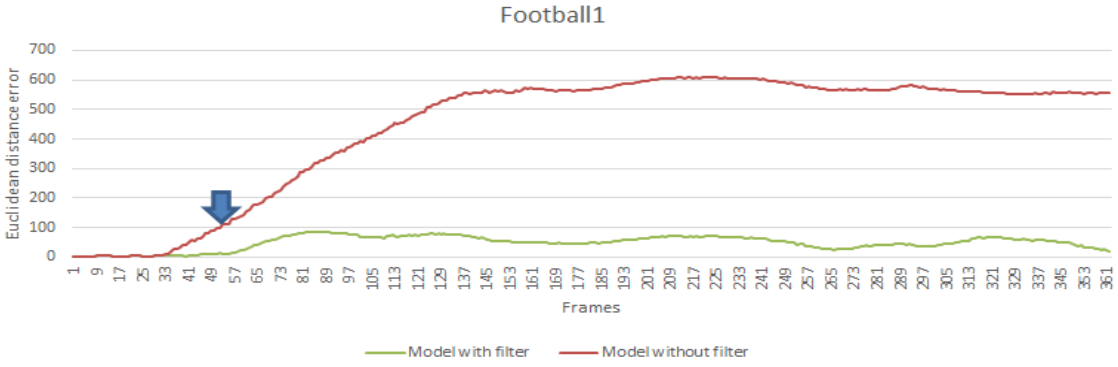


Figure 4.35: Football1 Euclidean distance comparison

4.6 Relative awareness

In this experiment we show the benefits of relative awareness as described in detail in Section 3.4.2.1. We use a very basic sequence of a child standing next to the target (abacus). We use the STC tracker to track the main target

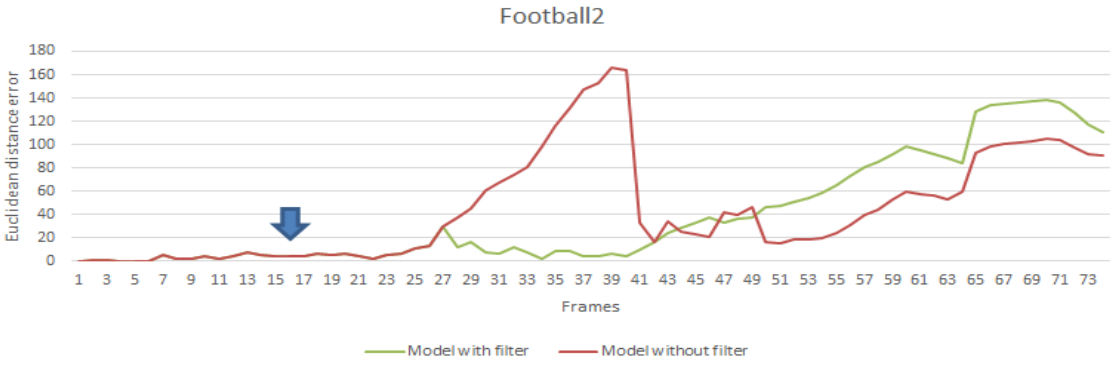


Figure 4.36: Football2 Euclidean distance comparison

and a face detector from Viola and Jones [25] to detect and manage the face of the child as a real-world supporter. The model gets trained for a few frames and then we deliberately disable the target tracker to help illustrate our point. Our model takes over using supporters from the background and the child to estimate where the target should be. The child then suddenly moves aggressively to a higher position.

With relative awareness disabled, the face supporter estimates to a wrong position for the target and can potentially sway the voting if the weighting for this real-world model is high enough, which in this case it was. See the yellow arrow in Figure 4.37, the red line is the contribution from the face supporter. With relative awareness enabled, the face of the child as a real-world supporter is not even considered active for voting due to the relative position it has with the target in comparison to where it was in the previous frame. In Figure 4.38 you will note there is no voting contribution from the face of the child we detected, only a few low-level SIFT supporters.

4.7 Stale supporters

This experiment focuses on the value of getting rid of stale supporters as described in Section 3.4.2.2. We took the same sequence as in Section 4.6. This time when our model is estimating the target position, we move the child from the field of view for more than five frames and bring her back at a higher position as per the previous experiment.

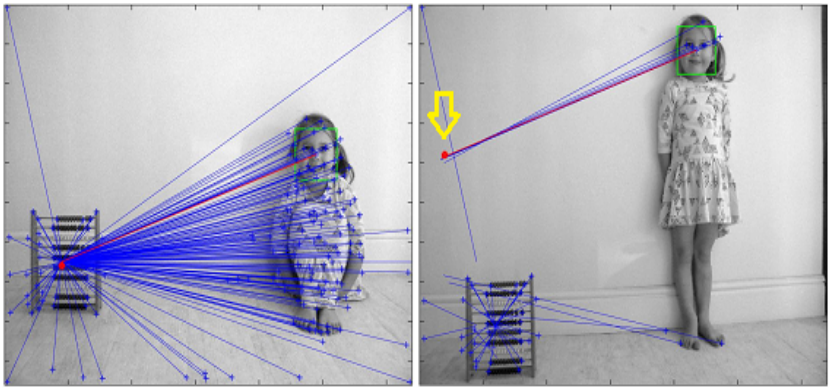


Figure 4.37: Relative awareness - disabled

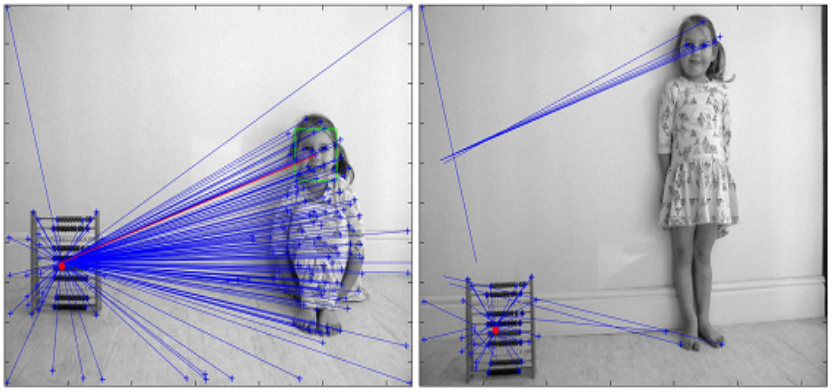


Figure 4.38: Relative awareness - enabled

Disabling the mechanism to get rid of stale supporters allowed for the outdated supporters on the face of the child, that showed up from nowhere, to effect the voting outcome, see Figure 4.39. Enabling the stale supporter mechanism discarded the supporters on the child as trained and trustworthy and instead saw them as new supporters ready to get trained as potential second-level supporters. See Figure 4.40.

4.8 Summary

Experimentation in this chapter shows the undeniable benefits of integrating context into the task of tracking. It is valuable information, and if used correctly can improve the task of tracking. The importance of correlated movement between context and the target was confirmed and present throughout every experiment. Particularly in the case of ***Red bull*** and ***Pedestrian***,

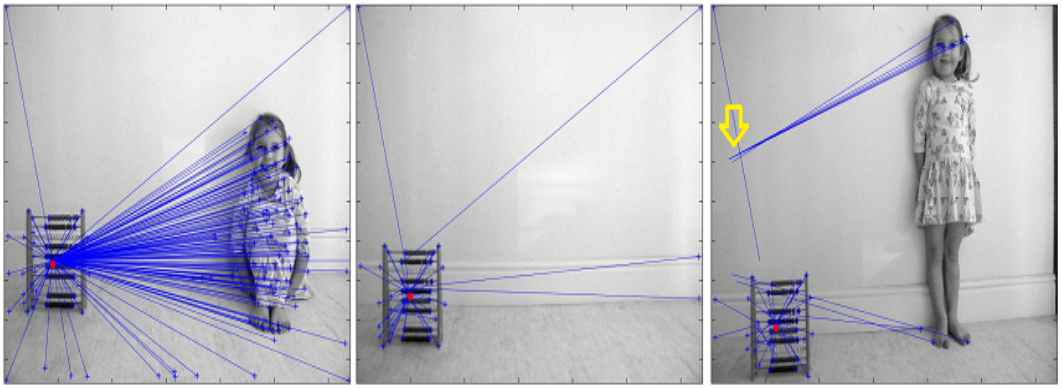


Figure 4.39: Managing stale supporters - disabled

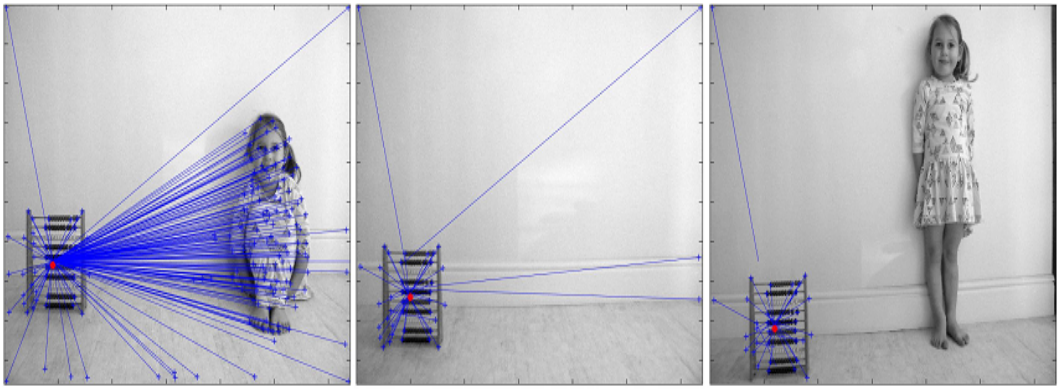


Figure 4.40: Managing stale supporter - enabled

in both these sequences the target is fully occluded and changes direction, in one case even leaving the FOV.

The biggest challenge as seen from experiments such as ***WalkingPerson***, ***FaceChange*** and ***Izandri***, is the problem of quantity over quality of supporters, especially in the event of sudden change in direction. Proper first-level supporters are sometimes overshadowed by many other good correlated supporters that are not so beneficial any more in the event of direction change. These are typically supporters that are part of the background or are static in nature and when the target suddenly changes direction can skew the voting.

Other challenges included limited correlated supporters as in ***WalkingPerson*** and ***Pillars***.

Most of these challenges can be overcome with the use of real-world supporters and the use of a priority filter, but there are still instances where the limited amount of correlated context, on which this model depends, is not available.

The benefits of good verification for the main tracker was highlighted in experiments ***Pedestrian*** and ***FaceChange***. In both cases the target tracker would have drifted to a false positive without the use of this useful mechanism.

In both the low-level and real-world supporter experiments, we show the ease of integrating off-the-shelf implementations for either the main target tracker or to help find real-world supporters. The model is context flexible, capable of integrating any form of context.

We showed near the end of this chapter the value of second-level supporters and how they can still add value even when the first-level supporters becomes extinct, giving more potential time for the main tracker to get back on track. During these particular experiments, the need for a Gaussian priority filter was revealed and implemented. See Section 3.4.6.

In Chapter 5 we look back at our original goals, the model we proposed, experimentations done and provide our final conclusions. We also focus on possible future directions and some recommendations.

Chapter 5

Conclusions

In our problem statement, (Section 1.3 on Page 6), we highlighted the challenges faced in tracking in an unconstrained environment. We decided to build/improve a model capable of tracking in such environments, using generative model techniques for on-the-fly learning and integrating knowledge (context).

Our model is built on the work of Grabner, and we selected this model for its simplicity, elegance and ability to integrate with other models. Our model is completely context flexible, capable of integrating any form of context.

The contributions made in Section 1.6 on Page 8, allowed for us to accomplish the goals we set out in Section 1.5 on Page 7, in particular the following three important goals:

- improve trustworthiness of generative models and to avoid drifting,
- improve on the methods integrating knowledge (context) by means of higher scene understanding and
- define a model/framework that easily integrates with and enhances existing tracking methods.

5.1 Limitations

As complementary to the task of tracking our model has proven to be, the training is heavily dependent on the success of the main target tracker. If the target tracker slowly drifts to a false positive, our model is trained on false information. Setting the threshold high for when to stop trusting the main tracker reduces this problem, but it remains a limitation.

The success of the model depends on having context that correlates well with the target, even just for a brief moment. Tracking a target in a relatively static background limits the use of valuable knowledge/context. Background knowledge can become very correlated with the target during training if the target remains static as well. When the target tracker fails and our model is active, this valuable background context becomes a problem when the target starts moving in a different direction to what it was trained on.

5.2 Recommendations and future directions

Ensuring proper matching techniques for supporters in subsequent frames provides for a more stable model. Although mechanisms like relative awareness and getting rid of stale supporters improve the probability of a successful match, having off-the-shelf implementations more capable in matching, ensures for a more confident model.

In our implementation SIFT features was considered the basic low-level supporter model, always present to drive mechanisms like relative awareness. The model could also depend on these low-level supporters in the event no real-world supporters are found. Having a generic class-less real-world supporter model for the same purposes provides for a good stand-alone implementation that can be used with any main target tracker. We integrated the work done by Alexe *et al.* [10] for this purpose, however it still requires work to be more reliable. Applying their algorithm over the same image twice, the boundary boxes found for potential objects still varies too much. This complicates the task of matching a supporter between frames

and requires a tracker.

The model can be improved to make clear distinction between background objects and potential moveable objects. As mentioned under limitations, background supporters are useful but they pollute the estimate when the target suddenly changes direction relative to the background. These once confident background supporters now becomes a burden. One could improve the model to detect when direction change occurs relative to the other supporters and then prioritise the objects that are categorised as non-stationary objects by nature. This ties into the concept of using knowledge not just for its positional properties, but the information on objects class, see Section 2.2 on Page 30. It is important to try and keep generalisation intact when exploring this direction. Perhaps classification should go as far as stationary and non-stationary objects with the latter getting priority when available.

Appendices

Appendix A

Implementation User Guide

In this section we provide information on the required software to run our implementation, as well as instruction for running the various test scenarios from the thesis.

A.1 Software used

The implementation was built and ran in Matlab R2015a (32-bit). The main algorithm will work on Matlab R2011b and later as well. In order to make use of the built-in detection libraries in some of the test scenarios, Matlab R2015a is required. To run the implementation on a 64-bit version of a particular operating system, some MEX files will have to be recompiled accordingly.

A.2 Running a test sequence

To execute our implementation and run specific test scenarios, the following steps are required:

1. Open Matlab and set the path to the following folder/module - **TrackingWithContext**.

2. In Matlab execute the following script - **TrackingWithContext**
   Example: » *TrackingWithContext*

3. A menu will appear with all the different dataset sequences available. Select a number.
   Example:
   Select a test sequence:
   1 - Redbull
   2 - Walking Person
   3 - Pedestrian
   4 - FaceChange
   5 - Gymnast
   6 - Izandri
   7 - Pillars
   n - ...

4. Based on the selection in the previous step you will be presented with a final list of options, containing detailed information on a particular testing scenario.
   Example of ***Walking Person*** detailed test scripts:
   Select a test script:
   1 - Full occlusion - Particle Filter - (100% SIFT)
   2 - Direction change - Particle Filter - (100% SIFT)
   3 - Direction change - Particle Filter - (40% SIFT | 60% Objectness)

5. If a test scenario completes successfully, an archive file will be produced in the current folder that contains a copy of each frame with its voting results illustrated, indicating where the target is. The archive also contains a file called **debug_output.xlsx** that contains the following information per frame processed:

   - **Frame number** - The frame number in the selected test sequence.

- **Model used** - Boolean value indicating whether our model provided the target position or if the main tracker was responsible.
- **Target X** - The x-axis position of the target.
- **Target Y** - The y-axis position of the target.
- **Ground X** - The x-axis position provided by a ground truth file loaded at the start. If not available this will be an empty cell.
- **Ground Y** - The y-axis position provided by a ground truth file loaded at the start. If not available this will be an empty cell.

The code and results from experimentation can be found in the following Dropbox folder: http://tiny.cc/e4ivey

List of References

[1] Hoiem, D., Efros, A. and Hebert, M.: Putting objects in perspective. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 2137–2144. IEEE, 2006.

[2] Yilmaz, A., Javed, O. and Shah, M.: Object tracking: A survey. *ACM Computing Surveys (CSUR)*, vol. 38, no. 4, p. 13, 2006.

[3] Yang, H., Shao, L., Zheng, F., Wang, L. and Song, Z.: Recent advances and trends in visual tracking: A review. *Neurocomputing*, 2011.

[4] Jalal, A.S. and Singh, V.: The state-of-the-art in visual object tracking. *Informatica (Slovenia)*, vol. 36, no. 3, pp. 227–248, 2012.

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

[6] Bishop, C. and Lasserre, J.: Generative or discriminative? getting the best of both worlds. *Bayesian Statistics*, vol. 8, pp. 3–23, 2007.

[7] Bishop, C.: *Pattern recognition and machine learning*, vol. 4. springer New York, 2006.

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

[9] Grabner, H., Matas, J., Van Gool, L. and Cattin, P.: Tracking the invisible: Learning where the object might be. In: *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1285–1292. IEEE, 2010.

[10] Alexe, B., Deselaers, T. and Ferrari, V.: Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2189–2202, 2012.

[11] Divvala, S., Hoiem, D., Hays, J., Efros, A. and Hebert, M.: An empirical study of context in object detection. In: *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009.*, pp. 1271–1278. Ieee, 2009.

[12] Yang, M., Wu, Y. and Hua, G.: Context-aware visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 7, pp. 1195–1209, 2009.

[13] Yang, M., Wu, Y. and Lao, S.: Intelligent collaborative tracking by mining auxiliary objects. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 697–704. IEEE, 2006.

[14] Zhang, K., Zhang, L., Yang, M.-H. and Zhang, D.: Fast tracking via spatio-temporal context learning. *arXiv preprint arXiv:1311.1939*, 2013.

[15] Li, Y. and Nevatia, R.: Key object driven multi-category object recognition, localization and tracking using spatio-temporal context. In: *Proceedings of the 10th European Conference on Computer Vision: Part IV*, pp. 409–422. Springer-Verlag, 2008.

[16] Leibe, B., Leonardis, A. and Schiele, B.: Combined object categorization and segmentation with an implicit shape model. In: *Workshop on statistical learning in computer vision, ECCV*, vol. 2, p. 7. 2004.

[17] Sun, Z., Yao, H., Zhang, S. and Sun, X.: Robust visual tracking via context objects computing. In: *2011 18th IEEE International Conference on Image Processing (ICIP)*, pp. 509–512. IEEE, 2011.

[18] Dinh, T., Vo, N. and Medioni, G.: Context tracker: Exploring supporters and distracters in unconstrained environments. In: *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1177–1184. IEEE, 2011.

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

[20] Helmut grabner. `http://www.vision.ee.ethz.ch/~hegrabne/DATA/ethCup.zip`. Accessed: 2016-08-23.

[21] Bonn benchmark on tracking. `http://www.iai.uni-bonn.de/~kleind/tracking/index.htm`. Accessed: 2015-08-08.

[22] Tracking learning detection. `http://personal.ee.surrey.ac.uk/Personal/Z.Kalal/TLD/TLD_dataset.ZIP`. Accessed: 2016-08-23.

[23] Tracking with online multiple instance learning. `http://vision.ucsd.edu/~bbabenko/data/miltrack/faceocc2.zip`. Accessed: 2016-08-16.

[24] Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping monte carlo sampling. `http://cv.snu.ac.kr/newhome/publication/dataset/BHMC_data.zip`. Accessed: 2016-08-23.

[25] Viola, P. and Jones, M.: Rapid object detection using a boosted cascade of simple features. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001. CVPR 2001.*, vol. 1, pp. I–511. IEEE, 2001.

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

[27] Online object tracking: A benchmark. `http://cvlab.hanyang.ac.kr/tracker_benchmark/seq/Jogging.zip`. Accessed: 2016-08-23.

[28] Online object tracking: A benchmark. `http://cvlab.hanyang.ac.kr/tracker_benchmark/seq/Football.zip`. Accessed: 2016-08-23.

[29] Online object tracking: A benchmark. `http://cvlab.hanyang.ac.kr/tracker_benchmark/seq/Football1.zip`. Accessed: 2016-08-23.