L-Università ta' Malta
Faculty of Engineering

SUPPLEMENTARY REPORT

DOCTOR OF PHILOSOPHY

Trajectory Space Factorisation for Vision-Based Automated Sign Language Recognition

---

Mark BORG

Supervised by
Prof Kenneth P. CAMILLERI

Advised by
Prof Marie ALEXANDER

A dissertation submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

by the

Faculty of Engineering

March, 2020

Copyright Notice

Contents

List of Figures

List of Tables

List of Acronyms

ARRSAC adaptive real-time random sample consensus.

ASLR automated sign language recognition.

CNN convolutional neural network.

EBM elliptical boundary model.

EM expectation maximisation.

FAST features from accelerated segment test.

FOV field of view.

GMM Gaussian mixture model.

GNN global nearest neighbour.

HOG histogram of gradients.

JPDA joint probabilistic data association.

KLT Kanade-Lucas-Tomasi.

LBP local binary patterns.

LMedS least median of squares.

MAP maximum a posteriori.

MHDA multiple hypothesis data association.

MHT multiple hypothesis tracking.

MLE maximum likelihood estimation.

MMSE minimum mean square error.

MoG mixture of Gaussians.

MOT multiple object tracking.

MSER maximally stable extremal region.

NGT Dutch sign language.

nRGB normalised RGB.

pdf probability density function.

pmf probability mass function.

PROSAC progressive sample consensus.

RANSAC random sample consensus.

RNN recurrent neural network.

SfM structure from motion.

SIFT scale-invariant feature transform.

SMC sequential Monte Carlo.

SRIF square root information filtering.

SURF speeded-up robust features.

SVM support vector machine.

Chapter 1

Introduction

This document contains supplementary material for the PhD dissertation:

- Mark Borg, "Trajectory Space Factorisation for Vision-Based Automated Sign Language Recognition", PhD thesis, University of Malta, 2020.

The rest of this document is structured as follows:

Chapter 2 provides more detailed descriptions of the algorithms and processes, used for signer body part detection in our automated sign language recognition (ASLR) pipeline, and introduced in §3 of the main PhD thesis report.

Then in **Chapter 3**, additional results are provided for the experiments carried out during the evaluation of these body part detection algorithms. The experiments are covered in §3 of the PhD thesis report.

In **Chapter 4**, more details on signing detection are provided (in addition to what is described in §4 of the PhD thesis report).

Chapter 5 provides a brief description of the Kalman filter – this is referenced a number of times in the main PhD thesis report, as well as in the final chapter of this document.

Finally, **Chapter 6** describes the multiple hypothesis tracking (MHT) algorithm in detail, its formulation, as well as covering a number of extensions to the standard algorithm. The material provided in this chapter adds more detail to what is covered in §§ 3 and 5 of the PhD thesis report.

Chapter 2

Signer Body Part Detection

This chapter provides more detailed descriptions of some of the algorithms and processes mentioned in §3 of the main PhD thesis report, titled "Body Part Tracking for Sign Language Recognition".

2.1 The Viola-Jones face detector

This section covers the Viola-Jones face detector [1, 2], first introduced in §3.1.1 of the main PhD thesis report. The Viola-Jones face detector makes use of a set of rectangular *Haar-like features*, each feature matching some simple visual property common to human faces (a *weak classifier*). The *AdaBoost* machine learning algorithm is then used to build a strong classifier from a linear weighted combination of such weak classifiers. Once trained, the cascade of filters is employed while scanning sub-windows of a video frame. Efficient detection is obtained due to the early rejection of many of the negative sub-windows (no face present), without having to evaluate all the features in the cascade [1, 2]. Please refer to Figure 2.1 for an illustration.

We can statistically interpret the Viola-Jones face detector as performing a hypothesis test at each image location and for a range of window sizes (multi-scale face detection), with the null hypothesis H_0 being that there is no real face and that the detector's response is due to noise, image artefacts, or background.

We make no particular assumptions as to the location and the size of the signer's face when applying the face detector. Thus the initial prior distribution for the Viola-Jones is taken to be uniform – assumption (A4) (main PhD thesis report, p. 35) only kicks in if we find multiple faces *after* running the face detector, and *only* when our ASLR system is starting up.

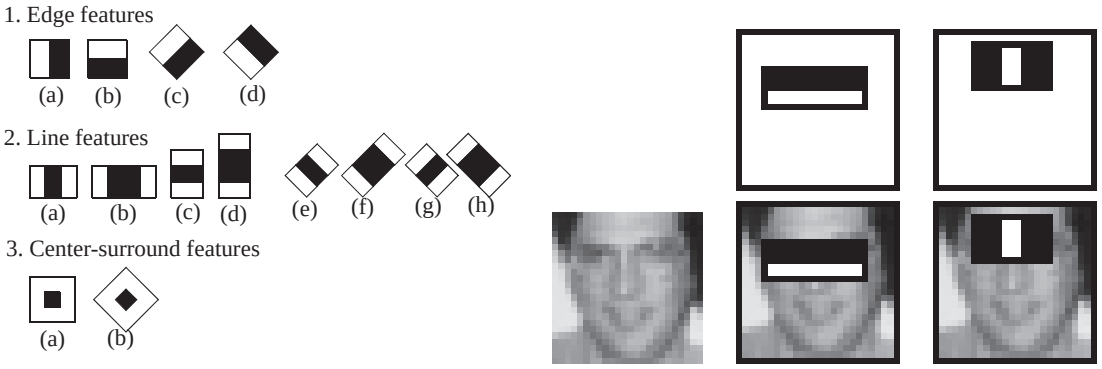


FIGURE 2.1: (Left) The Haar-like features of Viola and Jones [1], with the addition of the tilted versions and centre-surround features (1c, 1d, 2b, 2d to 2h, 3a and 3b) of Lienhart et al. [3]. Black areas have negative weights, while white areas are positive. When positioned on an image area (window), the Haar features compute the intensity difference between the black and white areas. (Source: Lienhart et al. [3]). (Right, top row) The first and second features in the cascade, as selected by the AdaBoost algorithm, shown overlaid on a face image (bottom row), indicating positions of highest response. (Source: Viola and Jones [2]).

We make use of Lienhart et al.'s [3] implementation of the Viola-Jones algorithm, which adds tilted versions of the Haar features to the standard set of the original algorithm, in order to improve detection. This implementation is available in the *OpenCV library* [4]. Haar-like features of size 20×20 are used, trained via a variant of AdaBoost, called *Gentle AdaBoost* [5]. We make use of two pre-trained Haar cascades that are provided in the OpenCV library, one for detecting frontal faces and the other for detecting profile faces. These two cascades together are able to cover the full range of out-of-image-plane face rotations ($\pm 90^\circ$ *yaw* movements of the head).

Additionally, we make use of 20°- and 40°-rotated versions of these Haar cascades [1] to handle large in-plane face rotations (head *roll* movements). Our adoption of 20° steps for in-plane rotation of the face is based on analysis on face detection errors versus rotation angles performed by Jones and Viola [6], Danisman and Bilasco [7].

We run these cascades in parallel, and perform decision fusion of the results, utilising the confidence score value τ given by Equation 3.1 (main PhD thesis report, p. 37). Depending on which Haar cascades are selected at decision fusion level, we can generate coarse estimates for the yaw and roll angles of the signer's head. Figure 2.2 illustrates the Viola-Jones multi-cascade face detector we employ in our ASLR system.

[1] The same effect can be achieved by rotating the input image clockwise and counter-clockwise by 20° and 40°, but this is computationally more expensive

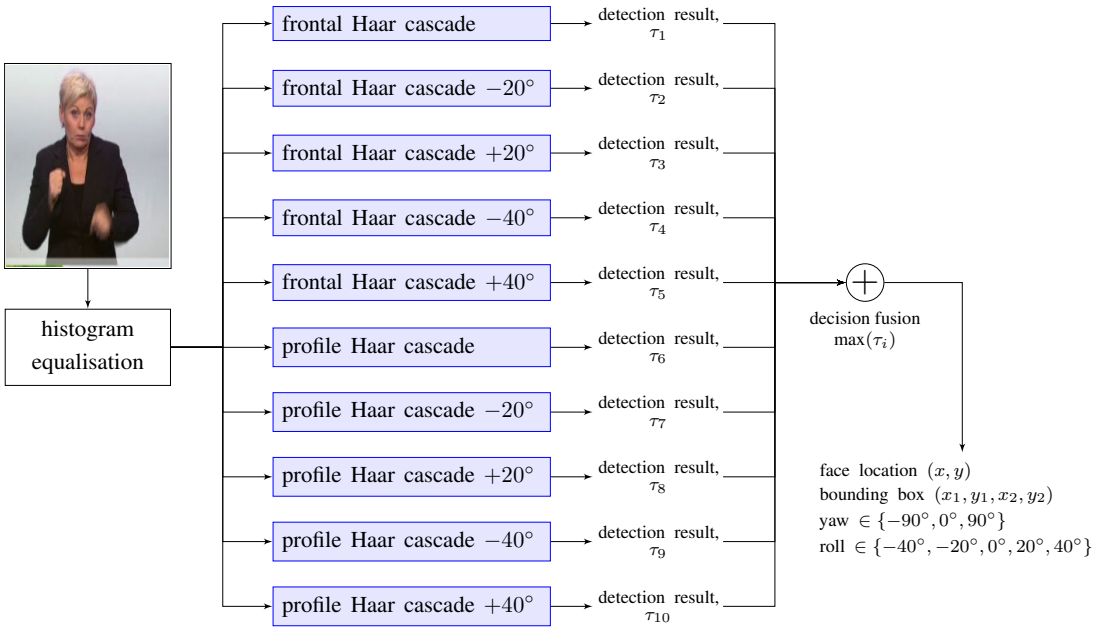


FIGURE 2.2: The Viola-Jones Face detector with multiple Haar cascades. Each cascade generates its face detection results and score τ, and then the maximal face detection response is determined via decision fusion. The output of the face detector consists of the signer's face location, face extent (expressed as a bounding box), and coarse estimates for head yaw and roll angles. The latter are determined based on the scores returned by the Haar cascades.

2.2 The MTCNN face detector

This section covers in a bit more detail the MTCNN face detector, first introduced in §3.1.1 of the main PhD thesis report (p. 37).

Zhang et al.'s [8] *Multitask convolutional neural network (MTCNN)* face detector uses three convolutional neural networks (CNNs) in a cascaded fashion. First, a fast and shallow CNN scans the entire image and generates multiple face region proposals (potential faces, several of which might be false positives). Each region proposal of the first CNN is then examined by two successive and increasingly more complex CNNs. These refine the results, by further filtering out false detections, performing non-maxima suppression, and applying more refinement of the bounding boxes of the remaining faces. The three neural networks are applied in a cascaded fashion for performance reasons.

Apart from the position and bounding box of a face, the MTCNN face detector also returns the positions of five major *facial landmarks*: eyes, nose, left and right mouth corners. It achieves this by employing a *multi-head* neural network architecture, consisting of three fully-connected output layers, one for each type of output (see Figure 2.3 for an illustration of the architecture).

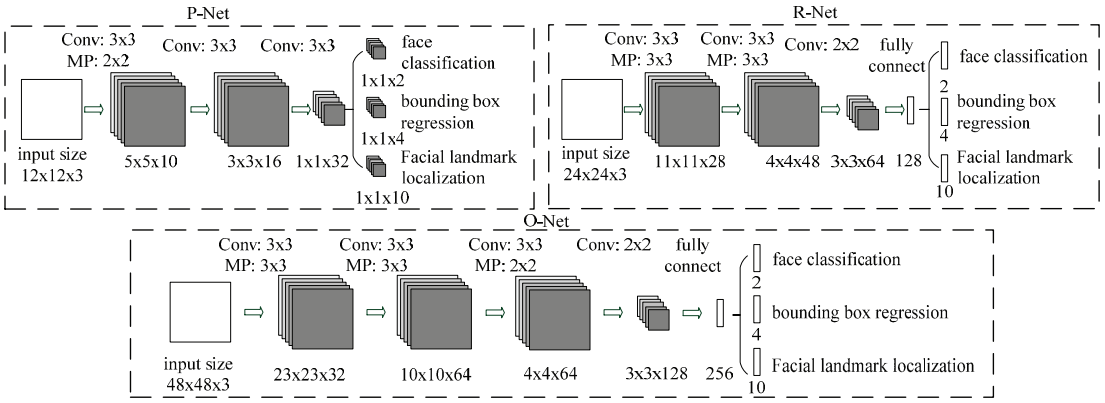


FIGURE 2.3: The architecture of the MTCNN face detector, consisting of a cascade of three CNNs of increasing depth and complexity: called P-Net, R-Net, and O-Net. All networks use convolution layers (denoted by 'Conv') and max pooling layers ('MP'). (Source: Zhang et al. [8]).

We employ a pre-trained version of MTCNN, implemented using the *Keras* and *TensorFlow* libraries [9, 10].

2.3 Upper Body Detection

This section gives some qualitative results for the investigations about upper body detection on signing videos that are mentioned in §3.1.4 of the main PhD thesis report.

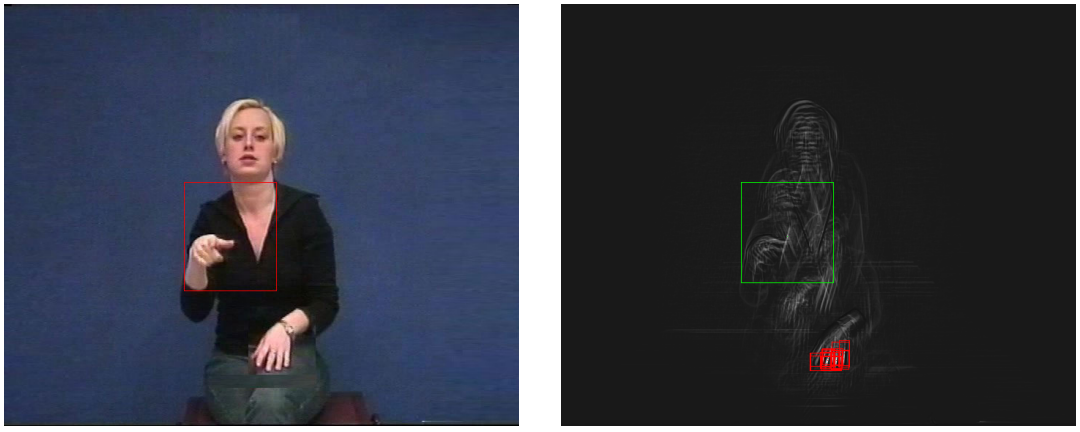


FIGURE 2.4: Results of running the symmetry detector of Dalitz et al. [11], while investigating upper body detection. (Left) Original video frame with detected regions of symmetry in red. (Right) The generated symmetry transform map showing maximal response. The strong response bounded by the green box is caused by reflective symmetry of the skin patches of the neck and the right hand of the signer. Strong responses are also given for the left hand due to repeated finger pattern. Note that we do not get any strong responses on the signer's midline.

In one of our experiments, we investigated whether symmetry detectors can be of help in upper body detection. We base this idea on the bilateral symmetry exhibited by the human body when viewed from a frontal or quasi-frontal viewpoint. We also rely on works in the literature that have reported successful results when using symmetry for pedestrian tracking [12].

One of the state-of-the-art symmetry detectors reported in the literature is that of Dalitz et al. [11]. Some qualitative results of this detector when applied to signing videos are given Figure 2.4. As can be seen, this detector fails to respond to the bilateral symmetry of the human body.

2.4 Skin Detection

This section gives additional background on the process of *skin detection*, first mentioned in §3.2.1 of the main PhD thesis report.

Skin detection is generally formulated as a pixel-level classification process, with colour serving as the main cue, since skin tones tend to occupy compact clusters in many colour spaces [13–20]. Adopting a statistical interpretation, the skin classification process assigns to each pixel a probability that its colour value, represented by the vector $\mathbf{c}$, can be explained by a skin colour model S, denoted by $P(\mathbf{c}|S)$.

In works like those of Gomez and Morales [21], a *non-parametric* skin colour model is used; for example via probability mass functions (pmfs) derived from histograms, or *skin probability maps*. For the former case $P(\mathbf{c}|S)$ is defined as:

$$P(\mathbf{c} = \mathbf{c}_i|S) = \frac{h_{skin}(\mathbf{c}_i)}{\sum_j h_{skin}(\mathbf{c}_j)} \tag{2.1}$$

where $h_{skin}(\mathbf{c}_i)$ is the histogram function returning the total number of colour values $\mathbf{c}_i$ encountered in the skin class.

Other works adopt a *parametric model*, such as a Gaussian model [15], a mixture of Gaussians (MoG) [13], or an elliptical boundary model (EBM) [22]. Fitting a skin colour model can then be formulated as a maximum likelihood estimation (MLE) problem:

$$S_{MLE} = \underset{S}{\operatorname{argmax}}\, P(\mathbf{c}|S) \tag{2.2}$$

and, the maximum likelihood estimators $\hat{\mu}$ and $\hat{\sigma^2}$ can be used, for example, to fit a Gaussian model; or the *expectation maximisation (EM) algorithm* can be used for fitting a MoG model.

By employing separate models for skin colours and non-skin colours (i.e, two-class classification rather than one-class), then the likelihood ratio test can be used for classification purposes:

$$\frac{P(\mathbf{c}|S)}{P(\mathbf{c}|\neg S)} \geq \lambda \tag{2.3}$$

where λ is some pre-determined discrimination threshold and $\neg S$ represents the non-skin colour model.

Given prior information about the probable occurrence of skin and non-skin colours in a video sequence, maximum a posteriori (MAP) estimation can be adopted instead of MLE. Then, using Bayes' rule:

$$S_{MAP} = \underset{S}{\operatorname{argmax}}\, P(\mathbf{c}|S)\, P(S) \tag{2.4}$$

$$\neg S_{MAP} = \underset{\neg S}{\operatorname{argmax}}\, P(\mathbf{c}|\neg S)\, P(\neg S) \tag{2.5}$$

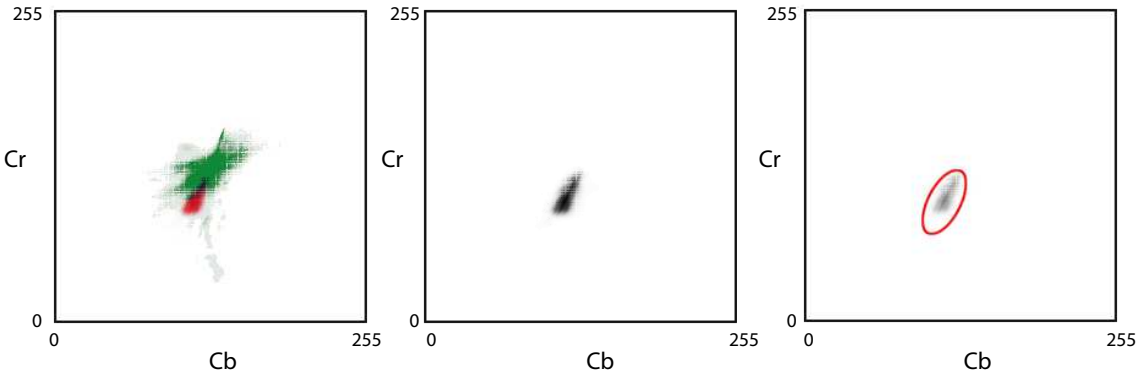
where the prior information $P(S)$ and $P(\neg S)$ can, for example, be determined from the total number of skin and non-skin pixels in the training data; or via prior information about where a person is located in the camera's field of view (FOV); or from previous evidence observed at some earlier time t.

As mentioned in §3.2.1 (main PhD thesis report, p. 44), when using a Gaussian parametric model, the assumption is that there is little overlap between skin and non-skin colours, and that a Gaussian is a good representation. But it is widely mentioned in the literature, that these assumptions are only approximately true [19, 23]. This is illustrated in Figure 2.5a, where it can be observed that skin colours do overlap non-skin colours [2]. And Figure 2.5c shows that a Gaussian can be a poor fit to the skin colour distribution [3]. Failure of these assumptions normally gives rise to a high level of false positives.

It is worth mentioning at this point that, apart from the statistically-based skin classifiers, there is also a large body of work that use empirically-defined threshold methods and/or fixed decision rules for skin detection [20, 21, 26, 27]. These are determined empirically in various colour spaces (e.g., RGB, HSV, YCbCr) after analysis of the skin-tone distributions in such spaces. One interesting work in this category is that of Gomez and Morales [21]: they empirically obtain a number of rules defining the

[2] These plots are using the chromatic components of the YCbCr colour space. Similar results are expected for any colour space, with the normalised RGB (nRGB) colour space we use in our skin classifier being no exception. Furthermore, Albiol et al. [24] provide a theoretical proof showing that the separability of skin and non-skin classes is independent of the chosen colour space.

[3] Refer to Elgammal et al. [25] for more example plots.



(A) Distribution of skin (red) and non-skin colours (green)

(B) Distribution of skin colours

(C) Gaussian model fitted to skin colours

FIGURE 2.5: The skin colour distribution in the YCbCr colour space for one of the videos in the BBC Pose dataset. Note the overlap between skin and non-skin in (A), and the non-Gaussian distribution in (B) and (C).

boundaries of skin colour regions within a colour space, and compare these rules against those obtained via decision trees.

Several research works [15, 17–19, 26] use an *adaptive* skin colour model. In contrast to a global or generic skin colour model, an adaptive model can be tuned to particular individuals, their skin complexions, ethnicity, etc. In addition, an adaptive skin model is in a better position to handle particular environmental conditions, like lighting variations. At each time step, Han et al. [18] incorporate updates derived from previously tracked skin regions into their skin Gaussian mixture model (GMM). Kawulok [17] uses an RGB statistical colour model updated dynamically based on detected face regions. The facial skin area is selected based on the inter-eye distance, after performing face detection followed by eye socket detection. Liensberger et al. [19] use skin pixels from detected faces, to adapt the skin colour model to the individual being tracked. In order to eliminate extraneous non-skin pixels at the outer edges of a face (like hair and background), the bounding box of the face region is reduced by a fixed size, and skin pixels are taken from within this smaller face area.

As mentioned earlier, pixel-based skin colour classification (where each pixel is classified independently of its neighbours), tends to generate a lot of false positives. While adaptive skin colour models can help to alleviate this problem, it is still a prevalent issue. This is not just limited to colour cues: Liensberger et al. [19] provide evidence demonstrating that skin detection in general, performed solely at the pixel-level, is an ill-posed problem.

Some works tackle this issue by performing classification on a *region-based level* rather than at pixel level, or by post-processing pixel-level classification with *region-growth operations* [18, 28–30]. For example, Deng and Manjunath [28] perform unsupervised segmentation of a video frame into separate spatial regions and then classify the regions as skin or non-skin. In the work of Han et al. [18], region-based information obtained via the JSEG algorithm is combined with pixel-level skin colour within a support vector machine (SVM) classification system. Wang et al. [29] employ region-based information obtained via shape and morphological operations. And Kawulok et al. [30] combine colour and texture features within a spatial analysis method that relies on identifying seed skin pixels and then using diffusion to grow the skin regions. Region-based skin segmentation typically provide more robustness, but at the expense of needing more computational power.

2.5 KLT feature tracking

This section delves a bit into the underlying principles and theory of the Kanade-Lucas-Tomasi (KLT) method, first introduced in §3.2.3 of the main PhD thesis report.

The KLT method knows its origin to the pioneering work of Lucas and Kanade [31], who tackled the image alignment problem, i.e., how to track (align) features from one video frame to the next. It was later enhanced by Tomasi and Kanade [32], and furthermore by Shi and Tomasi [33], on how to find features that are considered to be "good features to track".

Despite that many other types of more advanced sparse features have become recently available (see Tuytelaars and Mikolajczyk [34], and Li et al. [35] for some good surveys), the use of KLT features has remained very popular, mainly due to their 'lightweight' nature, good accuracy, and computational efficiency [4].

2.5.1 Optical flow

The KLT method is based on the notion of *optical flow*, the perceived apparent motion of the brightness patterns in an image, and what this can tell us about the true motions that are happening in the scene (i.e., the *motion field*; the true 3D motions as projected onto

[4] Govender [36] compared KLT features, Harris corners, speeded-up robust features (SURF) and scale-invariant feature transform (SIFT) features with regards to their impact on the accuracy of the structure from motion (SfM) problem. Her results show that amongst these four features, SIFT and KLT offered the best accuracy when it comes to the recovery of the camera's translation and rotation vectors. In terms of reprojection accuracy, while SIFT had the lowest reprojection error, KLT came second.

the 2D image plane) [37]. But in order to get meaningful motion information from the brightness changes visible in an image, the following key *optical flow assumptions* must hold over a small region (a local neighbourhood):

(KA1) The *spatial coherence assumption*: neighbouring points in the scene typically belong to the same surface and hence will have similar motions.

(KA2) The *temporal persistence assumption*: the image motion of a surface patch changes gradually over time.

(KA3) The *brightness constancy assumption*: illumination must be uniform (or changing in a known way), with no specular reflections, so that brightness changes in the image must be due to scene motion only.

Assumption (KA1) implies that a moving object should not exhibit any discontinuities or occlusion effects. Furthermore, objects in the scene behave rigidly (at least on a local neighbourhood) and do not undergo contractions, expansions and other forms of deformations [38]. It follows from (KA2) that points do not move very far from one time instant to another, i.e. they remain nearby. Assumption (KA3) can be formulated in terms of the following *brightness constancy equation*:

$$I(x, y, t-1) \;=\; I(x+u, y+v, t) \tag{2.6}$$

where the brightness at position (x, y) and at time $t-1$ is equal to the same brightness at a nearby position $(x+u, y+v)$ and a short time t later. Since the displacement (u, v) is assumed to be small (KA2), we can approximate it with a linear motion. And after applying a first-order Taylor approximation to the right-hand-side of Equation 2.6, we get the standard *optical flow equation* [37]:

$$I_x u + I_y v = -I_t \tag{2.7}$$

where $I_x = \frac{\partial I}{\partial x}$, similarly $I_y = \frac{\partial I}{\partial y}$, and $I_t = \frac{\partial I}{\partial t}$.

The main insight we get from Equation 2.7 is the following: the unknown displacement (u, v) is not determined by trying to explicitly match a point (x, y) in image at time $t-1$ with a nearby point in image t (as per Equation 2.6). Instead, we estimate in which direction the point has moved, by considering the local intensity changes (gradients I_x and I_y) within the image at time t. And we also take into account the change in brightness I_t from the image at time $t-1$ to the image at time t, at the same position (x, y) in both images. This is illustrated in Figure 2.6.

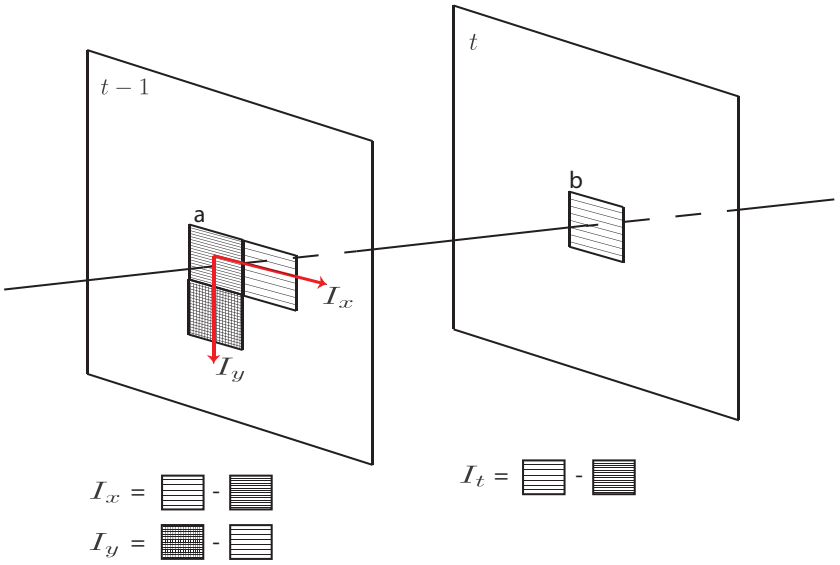


FIGURE 2.6: An illustration of the optical flow principle for the KLT algorithm, where the variation in brightness I_t at pixel location (x, y) from time $t-1$ to time t, matches the variations in brightness I_x and I_y within frame $t-1$ along the x and y axes. In other words, given a brightness increase from `a` to `b`, and knowing that there was a bright area to the right of `a`, one can conclude that the bright area moved leftwards by some amount. And the gradients I_x and I_y can determine the displacement needed to explain the brightness difference `b` $-$ `a`.

2.5.2 The aperture problem and neighbourhoods

Equation 2.7 is underconstrained (1 equation, 2 unknowns), giving rise to the *aperture problem*: one can only determine motion along one component, along the gradient direction (the so-called *normal flow*). A way to solve this is to add more constraints by roping in neighbouring points over a small area.

Dense optical flow methods like that of Horn and Schunck [39], make use of assumption (KA1) of page 10, which says that these neighbouring points must undergo a 'similar' displacement. Together with the addition of a regularisation term to enforce smoothness, such methods allow for the recovery of the displacement for every pixel in the image. On the other hand, while the *sparse optical flow* method of Lucas and Kanade [31] also relies on (KA1), they instead adopt the stricter interpretation that neighbouring points must have exactly the 'same' motion (u, v), and not just 'similar' motion.

Formulating the optical flow equation for a neighbourhood is done as follows: First, Equation 2.7 needs to be written in vector form:

$$\underbrace{[I_x \; I_y]}_{\nabla I} [u \; v]^T = -I_t \tag{2.8}$$

Then applying Equation 2.8 to a window neighbourhood of size $N \times N$, and aggregating over the set of constraints introduced by each of the pixels, we get:

$$\underbrace{\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ & \vdots \\ I_x(p_{N^2}) & I_y(p_{N^2}) \end{bmatrix}}_{\nabla \mathbf{I}} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{\mathbf{d}} = - \underbrace{\begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{N^2}) \end{bmatrix}}_{\mathbf{I_t}} \tag{2.9}$$

where p_i is the position of the i^{th} pixel in the window. This is now an *overdetermined system* (N^2 equations, 2 unknowns), and a *least-squares approach* can be used to find an approximate solution for the unknown displacement (u, v) of Equation 2.9:

$$\begin{aligned} & \min_{\mathbf{d}} \|\nabla \mathbf{I}\, \mathbf{d} - (-\mathbf{I_t})\|_2^2 \\ \implies & \left(\nabla \mathbf{I}^T \nabla \mathbf{I}\right) \mathbf{d} = -\nabla \mathbf{I}^T \mathbf{I_t} \\ & \mathbf{d} = -\underbrace{\left(\nabla \mathbf{I}^T \nabla \mathbf{I}\right)^{-1}}_{\mathbf{H}^{-1}} \nabla \mathbf{I}^T \mathbf{I_t} \end{aligned} \tag{2.10}$$

where $\mathbf{H}$ is the Gauss-Newton approximation to the true Hessian matrix [5]:

$$\mathbf{H} = \begin{bmatrix} \sum_p I_x I_x & \sum_p I_x I_y \\ \sum_p I_x I_y & \sum_p I_y I_y \end{bmatrix} \tag{2.11}$$

Equation 2.10 is a non-linear minimisation task. To solve this, Lucas and Kanade [31] use a *Newton-Raphson gradient descent* method: given an initial estimate for displacement $\mathbf{d}$, solutions are computed for increments $\Delta\mathbf{d}$ in an iterative manner ($\mathbf{d} \leftarrow \mathbf{d} + \Delta\mathbf{d}$) until the process converges to some required threshold ϵ:

$$\|\Delta \mathbf{d}\| \leq \epsilon \tag{2.12}$$

The stability of gradient descent depends on the inverse of the Hessian $\mathbf{H}$. This is stable whenever $\mathbf{H}$ is well conditioned, and not close to singular, i.e., when the eigenvalues λ_1

[5] The true Hessian is the matrix of second order derivatives. The Gauss-Newton approximation of the Hessian is $\mathbf{H} \approx \left(\nabla \mathbf{I}^T \nabla \mathbf{I}\right)$, where $\nabla \mathbf{I}$ is the Jacobian matrix of Equation 2.9. Note that $\mathbf{H}$ is a 2×2 matrix and the summations in Equation 2.11 are over all pixels p of the neighbourhood window.

and λ_2 of $\mathbf{H}$ obey the following conditions:

$$\lambda_1 \gg 0, \lambda_2 \gg 0, \min(\lambda_1, \lambda_2) > \xi \quad \text{eigenvalues are large \& above a threshold } \xi \tag{2.13}$$

$$\frac{\lambda_{max}}{\lambda_{min}} \approx 1 \quad \text{the eigenvalues have similar values} \tag{2.14}$$

2.5.3 Of affine warps and feature quality

While Equations 2.6 to 2.11 consider a displacement $\mathbf{d} = [u\ v]^T$ as the 'transformation' (movement) between two consecutive frames, the sparse optical flow method of Lucas and Kanade [31] can be generalised to handle any type of linear 'transformation' (for example, an affine warp). Letting $\mathbf{W}(x, y; \mathbf{d})$ denote a generic transformation (warp), with $\mathbf{d}$ being the parameter vector defining the warp, then Equation 2.6 becomes:

$$I(x, y, t-1) = I(\mathbf{W}(x, y; \mathbf{d}), t) \tag{2.15}$$

And Equations 2.9, 2.10 and 2.11 generalise to:

$$\nabla\mathbf{I} \frac{\partial \mathbf{W}}{\partial \mathbf{d}} \mathbf{d} = -\mathbf{I}_t \tag{2.16}$$

$$\mathbf{d} = -\underbrace{\left(\left[\nabla\mathbf{I}\frac{\partial\mathbf{W}}{\partial\mathbf{d}}\right]^T \left[\nabla\mathbf{I}\frac{\partial\mathbf{W}}{\partial\mathbf{d}}\right]\right)^{-1}}_{\mathbf{H}^{-1}} \left(\nabla\mathbf{I}\frac{\partial\mathbf{W}}{\partial\mathbf{d}}\right)^T \mathbf{I_t} \tag{2.17}$$

$$\mathbf{H} = \sum_p \left(\left[\nabla\mathbf{I}\frac{\partial\mathbf{W}}{\partial\mathbf{d}}\right]^T \left[\nabla\mathbf{I}\frac{\partial\mathbf{W}}{\partial\mathbf{d}}\right]\right) \tag{2.18}$$

respectively, where $\frac{\partial \mathbf{W}}{\partial \mathbf{d}}$ is the Jacobian of the warp matrix [6, 7] with respect to its parameters [40].

---

[6] For the translational case, warp $\mathbf{W}(x, y; \mathbf{d})$ is defined via the transformation matrix $\begin{bmatrix} 1 & 0 & u \\ 0 & 1 & v \end{bmatrix}$, where warp parameters $\mathbf{d} = [u\ v]^T$. Thus, $\begin{bmatrix} 1 & 0 & u \\ 0 & 1 & v \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x+u \\ y+v \end{bmatrix}$. Then $\frac{\partial \mathbf{W}}{\partial \mathbf{d}} = \frac{\partial \begin{bmatrix} x+u \\ y+v \end{bmatrix}}{\partial \mathbf{d}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ which is the identity matrix. Substituting the identity matrix $\mathbf{I_2}$ for $\frac{\partial \mathbf{W}}{\partial \mathbf{d}}$ in Equations 2.16, 2.17 and 2.18 makes them equivalent to Equations 2.9, 2.10 and 2.11.

[7] For the affine case, $\mathbf{W}(x, y; \mathbf{d}) \triangleq \begin{bmatrix} a_1 & a_2 & u \\ a_3 & a_4 & v \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 x + a_2 y + u \\ a_3 x + a_4 y + v \end{bmatrix}$, with $\mathbf{d} = [a_1\ a_2\ u\ a_3\ a_4\ v]^T$. Thus $\frac{\partial \mathbf{W}}{\partial \mathbf{d}} = \begin{bmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{bmatrix}$, and the Hessian is the 6×6 matrix $\mathbf{H} =$

The works of Tomasi and Kanade [32] and Shi and Tomasi [33] focus on the selection of the *sparse features* to be tracked across consecutive frames. Shi and Tomasi [33] adopt a pure translational model for tracking features across frames, i.e., they use the Lucas and Kanade [31] sparse optical flow method we have just covered and defined by Equations 2.8 to 2.11 in order to recover the displacement $\mathbf{d} = [u\ v]^T$.

While the features are being tracked from one frame to the next, any tracking errors and failures of the optical flow assumptions (KA1) to (KA3) given on page 10, will contribute to a gradual degradation of the tracking results. Shi and Tomasi [33] monitor the quality of a feature being tracked by examining its appearance in the surrounding window – if this suffers from a large change, then that feature is abandoned.

Shi and Tomasi [33] propose to use an affine deformation model to represent image changes, by setting the warp $\mathbf{W}$ in Equation 2.15 to an affine transformation matrix similar to the one given in Footnote 7 on page 13. While a feature is being tracked, the root-mean-squared intensity difference is computed between the current affine-warped window and the original window of when the feature was first created. The reason for comparing the current window with the first one, rather than consecutive windows, is to help minimise the *problem of drift* [32]. Thus, Shi and Tomasi [33] observe that while a translational model is sufficiently accurate to "track" features across consecutive frames (from $t-1$ to t), an affine model is more accurate for monitoring features across larger displacements (from t_0 to t).

2.5.4 Good features to track

As mentioned earlier, the stability of Lucas and Kanade's optical flow method [31], depends on the availability of the inverse of the Hessian in Equation 2.10. According to Shi and Tomasi [33], a "good feature to track" is one whose Hessian matrix $\mathbf{H}$ obeys the conditions stated in Equations 2.13 and 2.14.

For the translational displacement model, it can be observed that the Hessian given by Equation 2.11 is equivalent to the structure tensor used by the *Harris corner detector* [41]. Therefore "good features to track" are corner points that have strong gradients along both the x and y axes within the local neighbourhood. This is assured when the level of 'texturedness' (the intensity variations) in the surrounding neighbourhood is high. Having strong gradients I_x and I_y also implies more robustness to the *aperture problem* (§2.5.2). Therefore a key insight of the KLT algorithm is that we can determine

$\begin{bmatrix} xI_x & yI_x & I_x & xI_y & yI_y & I_y \end{bmatrix}^T \begin{bmatrix} xI_x & yI_x & I_x & xI_y & yI_y & I_y \end{bmatrix}$. This is not the only way how an affine warp can be parametrised; there are other ways, e.g., see Baker and Matthews [40].

that a corner is a good feature to track from just a single frame, based on the intensity variations (gradients) in the given image [40].

2.5.5 KLT window size and pyramidal KLT

The choice of window size in the KLT algorithm is typically a trade-off between, on one hand, having a larger window that is less sensitive to noise (more equations in Equation 2.9; stronger texture), and on the other hand, being small enough so that all the window pixels belong to the same surface (not straddling surface discontinuities), and being less likely to be affected by distortions due to changes in viewpoint, lighting, or non-rigidity [42].

The 'small motion' assumption (KA2) of page 10, often fails when the displacement between consecutive frames is larger than the chosen window size. Works like that of Bouguet [42] deal with this problem by proposing a *multi-resolution approach* for the tracking of the KLT features. First, a Gaussian pyramid is constructed via low-pass filtering and downsampling of each image frame. Then a hierarchical coarse-to-fine search technique is used by Bouguet [42], starting at the highest level of the image pyramid (the one with lowest details) and working down to lower levels (having finer details) and refining the KLT tracking results in the process.

Because of the multi-resolution element of the *pyramidal KLT algorithm*, the 'small motion' assumption (KA2) is more likely to remain valid: a large motion on the original image size that goes outside the KLT window, will exhibit a 'smaller' movement at a higher pyramid level, and thus is more likely to remain within the KLT window of that level. The result is improved tracking quality and longer lifetimes of the KLT features.

2.5.6 Assumptions of the KLT algorithm

Most tracking failures experienced by the KLT algorithm arise because of a failure of one or more of the underlying assumptions. Potential causes include: brightness constancy not satisfied (KA3), motion of a feature is not small (KA2), and a point does not move like its neighbours (KA1). The latter could be because the chosen window size is too large, or because the feature window straddles a depth discontinuity. Object occlusion can give rise to a lot of depth discontinuities in the scene. The KLT algorithm also does not work very well if an object changes shape, for example, the signer's hand undergoing a lot of deformations. Other potential sources of errors include image noise, defocus, and motion blur. The latter is also quite prevalent in signing videos.

Apart from the general optical flow assumptions of §2.5.1, other implicit assumptions of the KLT algorithm are:

(KA4) We assume square windows for the KLT features. And that all the pixels in a window (the neighbourhood) are similar to the feature point and contribute equally to the solution.

(KA5) It is assumed that a valid initial estimate for the motion parameters $\mathbf{d}$ in Equation 2.17 is available.

(KA6) The inverse of the Hessian of Equation 2.11 is well-conditioned and non singular.

Assumption (KA4) may not hold in general: pixels near the feature point are more likely to have a similar motion than those window pixels that are farther away. A possible way to mitigate this restriction is to use a Gaussian window rather than a uniformly-weighted square window to give more importance to pixels near the window centre [43]. But to our knowledge, all implementations of the KLT algorithm opt for the simpler square windows [4, 42, 44].

Assumption (KA5) must hold to ensure that the Newton-Raphson gradient descent method converges to the global minimum. As mentioned earlier, the Newton-Raphson method starts with the initial estimate for $\mathbf{d}$ (the last known position at time t–1), and then performs incremental parameter updates ($\mathbf{d} \leftarrow \mathbf{d} + \Delta\mathbf{d}$) iteratively until convergence is achieved. The parameter updates $\Delta\mathbf{d}$ are determined by $\nabla\mathbf{I}\frac{\partial\mathbf{W}}{\partial\mathbf{d}}$ which defines the direction of steepest descent [40]. If, amongst others, this estimate happens to fall outside the feature's window, because of a failure of the 'small motion' assumption (KA2), then convergence to an incorrect minimum occurs. This can result in *feature correspondence errors* (false matches). Using the pyramidal version [42] of the KLT algorithm can help to mitigate such a problem.

2.5.7 Handling the uncertainty of the KLT algorithm

A number of works have looked at how to model the uncertainty of the KLT tracker. For example, Ramakrishnan et al. [45] use the number of iterations taken by the Newton-Raphson gradient descent till convergence, as a very crude indicator of the quality of a feature track. This quality indicator is then used to vary the window size dynamically, so that it can adapt to distortions around the feature point in scenarios exhibiting large camera motions.

We found Ramakrishnan et al.'s approach [45] not to work in our case. For computational efficiency reasons, the KLT implementation we use [4], places a limit on the number of iterations of the Newton-Raphson gradient descent. From our investigations,

we found that for a majority of KLT features, convergence was not happening before hitting the iteration limit, thus rendering the proposed quality indicator ineffective.

A different approach is taken by Kanazawa and Kanatani [46]. They show that the uncertainty of a feature's position depends on the 'curvature' of its local intensity, and use the inverse of the Hessian ($\mathbf{H}^{-1}$) as an estimate for the 2D error covariance matrix. Similarly, Sun et al. [47] also make use of the inverse of the Hessian, and show that it gives a good approximation of the real error covariance matrix but up to a scale factor.

Note that the above works [45–48] focus on errors in KLT tracking caused by uncertainty and drift; they ignore feature correspondence errors (false matches). This is typical for studies performing *covariance propagation analysis*, where the focus is on modelling the uncertainty of a KLT feature as it is successfully tracked from frame to frame, and analysing whether the uncertainty in the feature's position is stable or grows with time.

For the case of feature correspondences errors, one way of dealing with such cases is to detect these as *outliers*, assuming that the number of successfully-tracked features outnumber the false matches. Commonly-used techniques for *outlier detection* include the random sample consensus (RANSAC) algorithm, an iterative outlier rejection method [49, 50]. Several variants of RANSAC are used for KLT-based tracking systems, including adaptive real-time random sample consensus (ARRSAC), and progressive sample consensus (PROSAC) [51, 52]. The latter utilise the KLT feature correspondence confidence score to accelerate the outlier rejection process. Apart from RANSAC and its variants, least median of squares (LMedS) is also commonly used for outlier detection [53].

An interesting approach is that of Wu et al. [54] and Hedborg et al. [55]. They detect outliers by utilising the *time reversibility* characteristic of the KLT algorithm – a feature is first tracked forwards in time (from frame t to $t+1$), and then tracked again backwards (from $t+1$ to t) by running KLT a second time. If the recovered motions of the two runs do not agree, then the feature is considered to be an outlier and is abandoned.

Sheorey et al. [43] look at the problem of KLT false matches from a theoretical perspective. They consider false matches as arising when the Newton-Raphson gradient descent converges to a local minimum. Such a situation is modelled with a GMM, with each Gaussian component representing the probability of the tracker getting stuck in a corresponding minimum. Then they derive a single covariance from the GMM to represent the overall uncertainty of a false match. While Sheorey et al. [43] perform a sound statistical analysis, implementing their method appears to be impractical, as it requires finding all local minima of the gradient descent.

Wong and Majji [56] utilise *sensitivity analysis* in order to propagate uncertainty of a tracked feature over time. They formulate the propagation of a feature's position on Equation 2.6, and represent uncertainty as additive Gaussian noise:

$$\mathbf{v} + \delta\mathbf{v} = (\mathbf{u} + \delta\mathbf{u}) + (\mathbf{d} + \delta\mathbf{d}) \tag{2.19}$$

where $\mathbf{u}$ and $\mathbf{v}$ are the previous and current feature positions respectively, $\mathbf{d}$ is the recovered displacement vector, and $\delta\mathbf{u} = \mathcal{N}(0, \Sigma_u)$, $\delta\mathbf{v} = \mathcal{N}(0, \Sigma_v)$, and $\delta\mathbf{d} = \mathcal{N}(0, \Sigma_d)$ are error terms.

Wong and Majji [56] demonstrate that the propagated uncertainty of a feature can only increase in time ($\Sigma_v \geq \Sigma_u \forall t$, since no *new* information about the KLT feature is coming in) – eventually this will violate the KLT tracker's assumptions. They propose using the length of the principal axis of covariance matrix Σ_v as a health metric of a feature point, testing it against a chosen threshold. Furthermore, they suggest that KLT features whose error ellipses (derived from the covariances Σ_v) intersect, have a higher chance of feature mismatch, and hence should be abandoned. One limitation of Wong and Majji's work [56] is that they do not specify how the initial value Σ_u can be estimated, i.e., when a KLT feature is first detected.

Modelling the errors and uncertainty of the KLT algorithm can prove very important for later components in our ASLR pipeline, in particular for the factorisation component.

2.5.8 Sub-pixel localisation of KLT features

While the standard KLT algorithm returns feature positions localised up to pixel-level accuracy, it is possible to further refine these positions in order to get sub-pixel level accuracy.

Sub-pixel localisation can be achieved by using the minimisation technique proposed by Sroba and Ravas [57]. Given $\mathbf{q}$ as the estimated position of the KLT feature (or corner), initially set to the pixel-level position of the feature returned by the KLT tracker, Sroba and Ravas observe that the vector from $\mathbf{q}$ to any point $\mathbf{p}_i$ in the feature's window W, is orthogonal to the gradient at $\mathbf{p}_i$. Sub-pixel localisation of $\mathbf{q}$ can thus be achieved by iteratively minimising the following dot product:

$$\epsilon = \sum_{\mathbf{p}_i \in W} \nabla I_{\mathbf{p}_i} \cdot (\mathbf{q} - \mathbf{p}_i) \tag{2.20}$$

The dot product in Equation 2.20 is zero when either (i) the gradient $\nabla I_{\mathbf{p}_i}$ is zero, (ii) when $\mathbf{q} - \mathbf{p}_i$ is zero, or (iii) the two vectors are perpendicular to each other. If point

$\mathbf{p}_i$ happens to fall on a flat region within window W, then case (i) applies and the dot product is zero. If point $\mathbf{p}_i$ falls on an edge in W, the vector $\mathbf{q} - \mathbf{p}_i$ must also fall on the same edge, since $\mathbf{q}$ is estimated to be at the corner point; thus by default case (iii) applies.

2.6 Grouping features into objects

This section gives additional background on our process of clustering sparse KLT features with common motion into objects. This process is first mentioned in §3.2.4 of the main PhD thesis report. We first briefly cover other works that adopt such an approach, some of which served as an inspiration for our work.

The works of Thirde et al. [58] and Borg et al. [59] bear the most resemblance to our work. They track KLT features from frame to frame, and then group them together using proximity and motion similarity to form candidate object observations. These are then matched to object tracks using joint probabilistic data association (JPDA), with the addition of track maintenance logic to handle occlusions, new objects, merging and splitting objects.

Another very similar work is that of Klicnar and Beran [60]. They iteratively fit motion models via a RANSAC-based algorithm to KLT features in each frame. The fitted motion models are assigned group labels, and then an object matching algorithm is used to link groups of features across frames. Matching is done via both forward and backward propagation of labels to increase robustness.

Sivic et al. [61, 62] adopt a similar approach. They make use of affine-invariant interest points and maximally stable extremal regions (MSERs) as their sparse features. Motion models are iteratively fitted to the features using the RANSAC algorithm, except that model fitting is done on consecutive groups of three frames, rather than just two.

Another similar work is that of Tanathong and Banharnsakun [63], who perform hierarchical clustering of KLT features in order to group them into feature blocks to be used in multi-object tracking.

Similarly, both Wong and Spetsakis [64] and Du and Piater [65] group spatially-close KLT features that have similar motions, but in this case using the EM algorithm for clustering.

In another similar work, Saunier and Sayed [66] cluster KLT features based on common motion and spatial proximity cues to track vehicles passing through road intersections. They represent the clustering as a graph structure, with feature tracks as the vertices, edges being grouping relationships, and connected components corresponding

to vehicle hypotheses. One limitation of graph-based clustering is that initially features tend to be 'overgrouped', but then the objects get segmented out (via graph disconnection) as more information becomes available.

Our work is also similar to that of Dimitriou et al. [67], in the use of motion models to describe the motion of clustered features. Dimitriou et al. [67] use point trajectories generated from optical flow as features. Spectral clustering is then used for grouping trajectories based on the similarity of the fitted motion models.

Less similar works include that of Buddubariki et al. [68], who locate objects via face detection, extract SURF features from the detected objects' areas, and then track the SURF features using a KLT tracker. To remove non-moving features belonging to the background, they convolve the magnitude of gradients with optical flow. A limitation of this work is that full object occlusion is not handled.

In the work of Bilinski et al. [69], features from accelerated segment test (FAST) with histogram of gradients (HOG) descriptors are used. Candidate objects are formed by grouping foreground connected components (determined via a background subtraction algorithm), and then associated with target objects using the motion and appearance information of the FAST features. Tracking through occlusion is performed by splitting the foreground region into objects according to their separate features.

Yu et al. [70] combine corner features with spatial features and colour features in order to track objects through cluttered scenes. To handle occlusion and object mergers, they use GMMs to represent colour information and corner features (with one GMM component per corner feature), as well as shape priors.

Byeon et al. [71] make use of a foreground map, KLT features and region features. The latter are obtained via tree-based MSERs. MSERs are stable homogeneous regions where edges and corners are rarely detected. They employ a hierarchical approach to grouping: first performing region-level grouping using motion similarity (average velocity) and foreground region overlap (computed via delaunay triangulation); then this is followed with object-level grouping, formulated as a minimisation problem and solved via the use of simulated annealing.

Kim [72] consider the challenging aspect when objects can have different scales (hence number of features and proximity can be problematic). They handle this by doing a dynamic multi-level grouping: individual features (Förstner corners) in foreground regions are grouped into small clusters, which in turn are grouped into object-level clusters. Clustering is done via the use of the EM algorithm.

And finally one of the earlier works, Wang and Adelson [73], group KLT features into affine motion layers using the k-means algorithm.

We note in passing, that there are many works that operate on the video sequence as a batch, performing 'cluster analysis' on full trajectories, e.g., Fradet et al. [74] and Zeppelzauer et al. [75]. These methods are not suitable for online operation, which is typically the case for ASLR.

2.7 Long-term hand tracking

This section provides some additional information on the topic of long-term hand tracking, first covered in §3.2.5 of the main PhD thesis report. In order to choose the tracker that works best for our ASLR system, we investigated a number of widely-available trackers, guided by the literature and by the particular requirements pertinent to signing. These trackers and some of their results are listed below:

The CMT tracker [76]. We used the C++ implementation of the CMT tracker, called *CppMT*, available here: https://github.com/gnebehay/CppMT. Figure 2.7 shows some results. This tracker performs quite well when tracking the signer's face. It shows

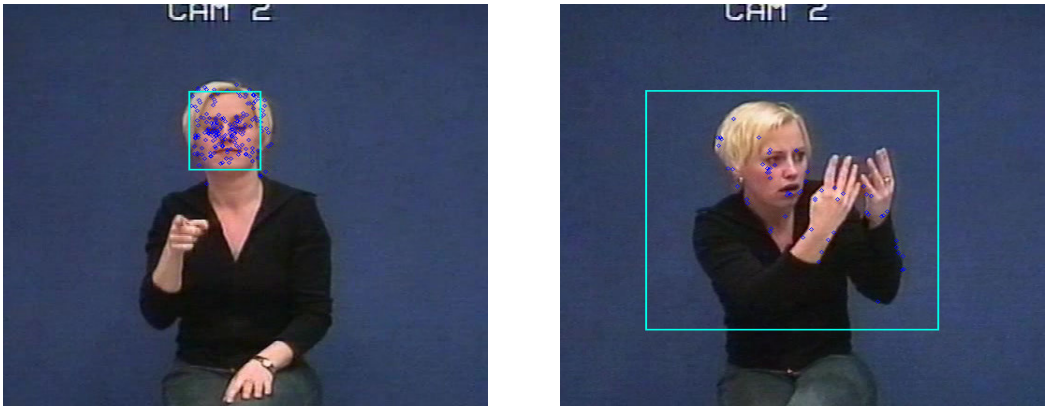


FIGURE 2.7: Results of the CMT tracker [76], showing correct behaviour (left) when tracking the face, and (right) a failure case when used to track the right hand.

robustness to viewpoint changes and object rotation. But its tracking degrades when the target undergoes occlusion. For the face, the CMT tracker manages to recover, mostly due to the fact that the face is only partially occluded. When applied to the tracking of the right hand, the tracker quickly drifts away from the hand to either the other hand, the face, or a combination of both (the tracker exhibits some stochastic behaviour and behaves differently on repeated runs). The hands change orientation rapidly and less features are detected and tracked in the hand region when compared to the face region.

The LSHT tracker [77]. We use the implementation at: www.shengfenghe.com/visual-tracking-via-locality-sensitive-histograms.html. Figure 2.8 shows some results. This tracker gives quite poor results: it suffers from frequent loss of the hands, when these undergo drastic and rapid deformations, and exhibits limited recovery from such situations.

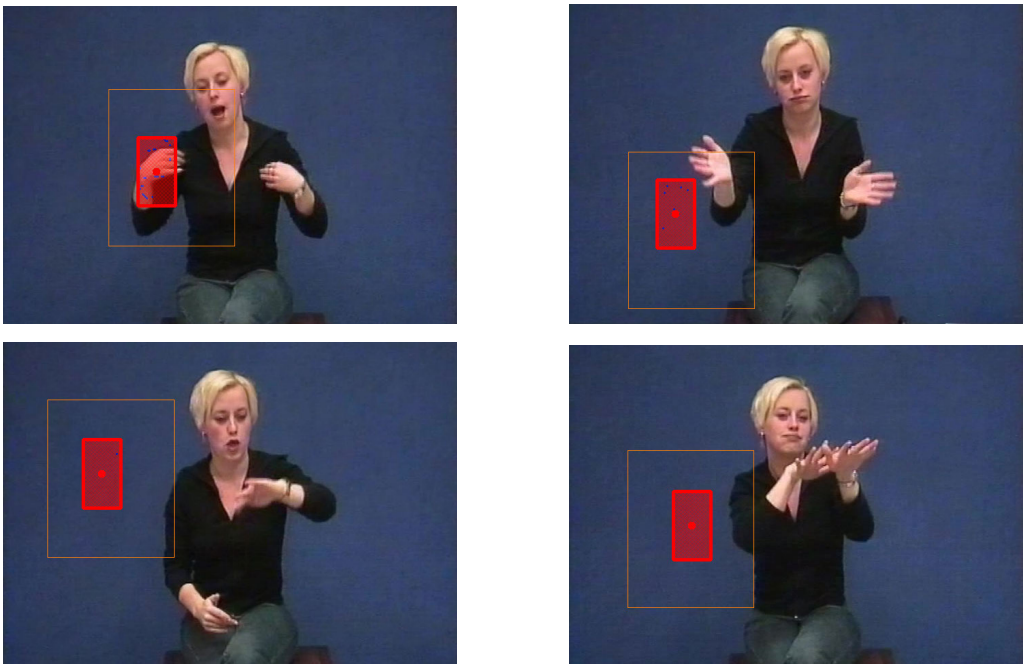


FIGURE 2.8: Results of the LSH tracker [77] showing the object drift problem, denoted by the red bounding box.

The HHT tracker [78]. This tracker is designed specifically for the tracking of the head and hands of a signer or speaker, and intended for video settings very similar to the ones we use. It is made available as part of the *HHI-VideoRecognizer plugin* for the *ELAN* linguistic annotation tool [79]. Access to this tracker can be found here: https://tla.mpi.nl/projects_info/auvis/, and the ELAN tool is available here: http://tla.mpi.nl/tools/tla-tools/elan. Figure 2.9 gives some results.

The MHT tracker [80]. We use an implementation of the MHT algorithm available here: https://github.com/WeatherGod/MHT. This version is in turn based on the original code developed by Cox and Hingorani [80], which is a C++ implementation of the standard MHT algorithm. This version was used for our initial investigations; later on, we developed our own implementation of the MHT algorithm, that also incorporates a number of enhancements mentioned in the literature and customisations for its use within our ASLR pipeline.

FIGURE 2.9: Results of the HHT tracker [78], illustrating good localisation of the head and hands, but an inability to track the objects successfully through occlusion.

The TLD tracker [81]. We use an implementation of the *Tracking, Learning and Detection (TLD)* algorithm that is provided with the *OpenCV library* [4]. This tracker exhibited a number of identity swaps between the right and left hands of the signer.

The KCF tracker [82, 83]. We use the *Kernelised Correlation Filter (KCF)* algorithm of Henriques et al. [82], with the extensions proposed by Danelljan et al. [83]. This implementation is provided with the OpenCV library [4]. The results of this tracker were quite poor: several instances of object drift, inability to track trough occlusions, etc.

The MIL tracker [84]. We use the *Multiple Instance Learning (MIL)* tracker that comes with the OpenCV library. This tracker gave quite poor results.

The GOTURN tracker [85]. We use the implementation of the *Generic Object Tracking Using Regression Networks (GOTURN)* that is provided with the OpenCV library. Figure 2.10 shows some results for this tracker, which can suffer from the loss of the objects when these undergo strong partial occlusions.

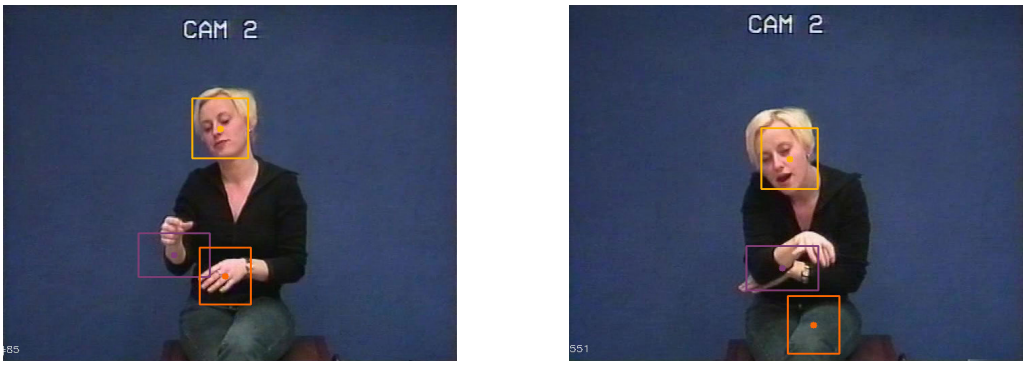


FIGURE 2.10: Results of the GOTURN tracker [85], showing the loss of the left hand when the hands merge together.

Chapter 3

Additional Results for Signer Body Part Detection

This chapter provides additional results for the experiments carried out in §3 of the main PhD thesis report, titled “Body Part Tracking for Sign Language Recognition”.

3.1 Face detection results

The results provided here are in addition to those presented in §B.1 of the main PhD thesis report.

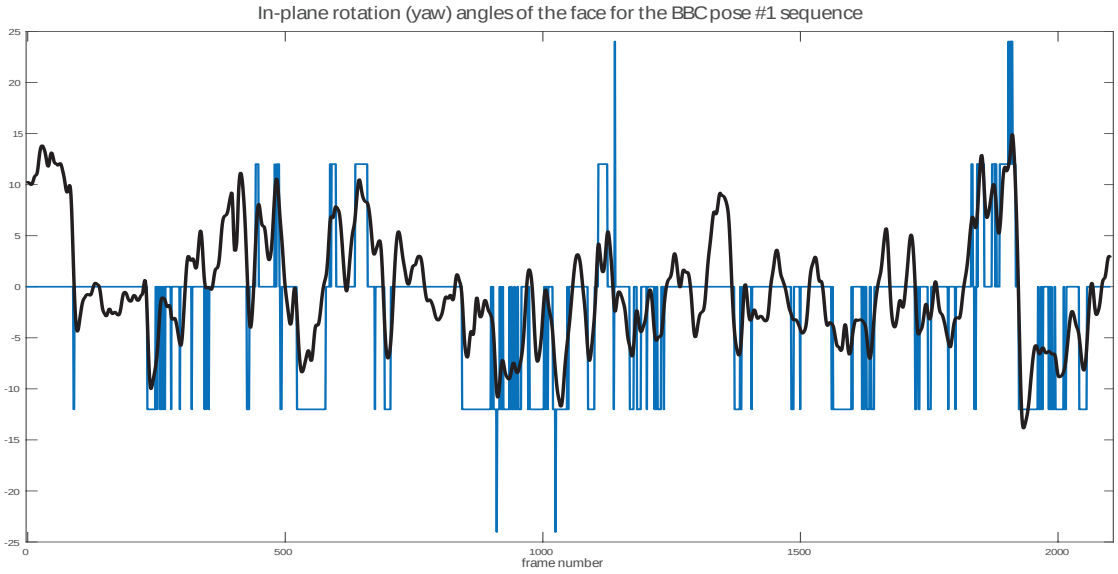


FIGURE 3.1: A comparison of the head in-plane rotation (yaw) angles returned by the Viola-Jones (in blue) and the MTCNN (in black) face detectors. The estimates generated by the Viola-Jones detector are very rough estimates with high granularity.

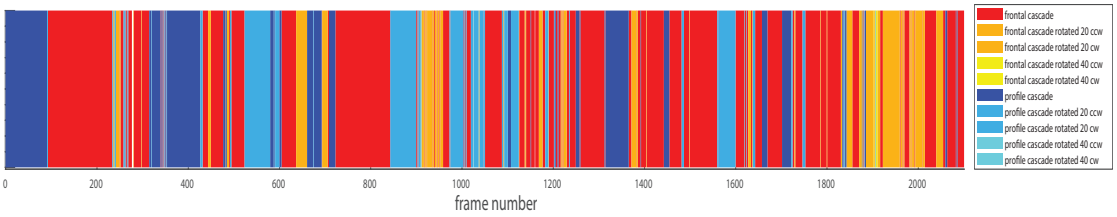


FIGURE 3.2: This time strip shows the Haar cascade with maximal response for the Viola Jones face detector for the BBC pose #1 sequence. This is used to estimate the head pose angles.

Figure 3.1 shows a comparison of the head rotation angles as returned by the two face detectors. The Viola-Jones estimates the head rotation angles from the Haar cascade which gives the maximal response, as described in §3.1.2 (main PhD thesis report, p. 37) and illustrated in Figure 3.2.

And Figures 3.3 and 3.4 show sample face detection results.

FIGURE 3.3: Sample results of the Viola-Jones face detector. Groundtruth is shown in green, while detections are depicted in red and yellow (for the frontal and profile Haar cascades respectively).

FIGURE 3.4: Sample results of the MTCNN face detector. Groundtruth is shown in green, while detections are in red. Also shown are the detected facial landmarks.

3.2 Face tracking results

The results provided here complement the ones presented in §B.2 of the main PhD thesis report.

In our implementation, we employ the observation's innovation (determined by the Kalman filter with respect to its prediction) as a rudimentary outlier rejection scheme [86]. We reject face detections whose innovation magnitudes do not satisfy the following condition:

$$|q_t| \leq \kappa \sqrt{\mathbf{S}_t} \tag{3.1}$$

where q_t is the innovation at time t, $\mathbf{S}_t$ is the innovation covariance matrix, and κ is a constant determining the inlier probability threshold. In effect this serves as the validation gate mechanism for the global nearest neighbour (GNN) tracker. Figure 3.5 shows outlier rejection results for the ECHO NGT_AH_fab1 sequence. We found out that this simple

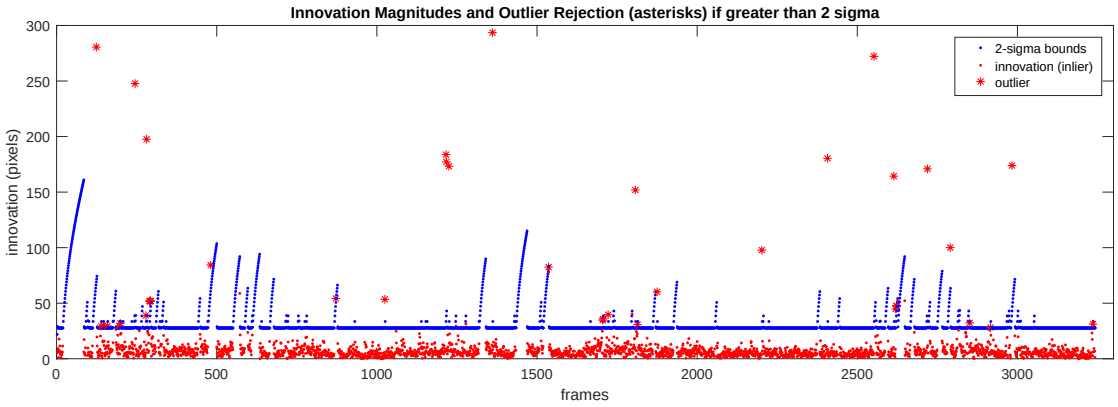


FIGURE 3.5: Innovation magnitudes of face observations are depicted in red, while the 2-sigma bound (serving as a validation gate) is shown in blue. Any face observations that have a magnitude greater than the bound are marked as outliers (red asterisks). Missing entries are due to mis-detections and face occlusion events. Note how the innovation bound increases as uncertainty increases when a face is not observed.

mechanism helps to minimise tracker drift, for example, when another person's face comes in the vicinity of the occluded signer's face.

Figure 3.6 shows sample frames for the output of the MHT tracker. Face observations obtained from the face detector are shown as black boxes, and the measurement noise associated with these observations are shown as black ellipses with dashed outline.

When the MHT tracker successfully associates and updates a target face with an observation, the target's state (position and bounding box) is shown as a dashed green box, while its state covariance (uncertainty) is shown as a dashed green ellipse. When the MHT tracker does not perform an association with an observation, then these face targets are shown in red.

It can be seen from Figure 3.6, that the MHT is able to correctly track the signer's face even in the presence of extensive occlusion and clutter (other nearby faces). Only one loss occurs over the shown time period and this is due to an occlusion event that exceeded the depth of the hypothesis tree of the MHT algorithm – when forced to make a hard decision, the MHT algorithm chose the hypothesis representing an object death rather than any of the other data association hypotheses with nearby clutter. We can help alleviate such problems, by using a larger tree depth, or incorporating additional logic to "stitch" face tracklets together.

FIGURE 3.6: MHT face tracking results for BBC Pose #16. Observations in black; Successful tracks in green; Estimated positions when unobserved are in red.

3.3 Skin and motion detection results

The section provides additional skin and motion detection results to the ones presented in §B.3 of the main PhD thesis report.

Figure 3.7, gives motion detection results for various datasets, while Figures 3.8 and 3.9 show skin detection results.

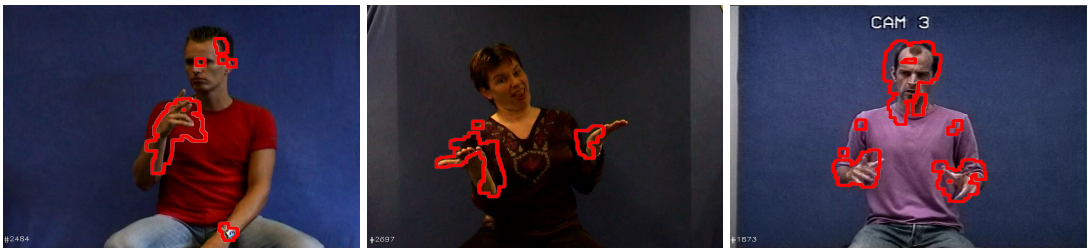
Sample frames in Figure 3.7 (B), (D) and (E) show significant camera motion, resulting in a large fraction of the video frame pixels being classified as motion. This negates the benefits that using a motion mask provides later on during sparse feature selection. Despite this, the majority of sign language videos are of a more static nature, and the hands can be discriminated well from the rest of the scene via their movements.

The standard frame differencing technique can suffer from the *hole problem* [87], resulting in the hands not being fully segmented out. In our experiments, we determine that using multi-frame differencing with a temporal window duration set to 10 frames, helps to alleviate this problem across a wide range of sign language videos.

We still observe cases where the signer's hands are not fully segmented out, especially when a hand is viewed edge on. Or the motion detector missing out on small finger structure – for example, see Figure 3.7 (A) left and centre.

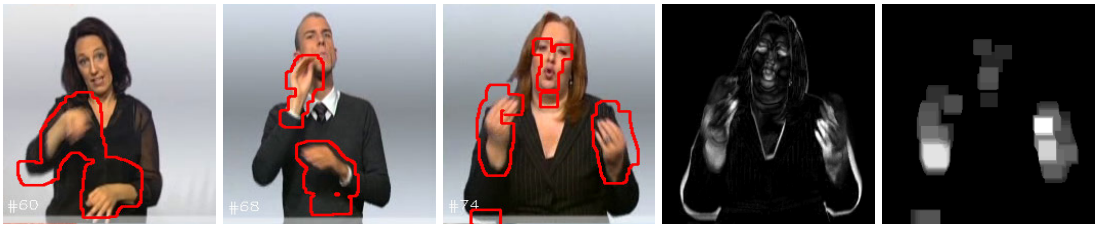
Figure 3.8 (C) and (D) right, show examples where, despite skin colour model adaptation and the fact that we are using both skin and non-skin colour models, parts of the clothes of the signer are still mis-labelled as skin. As described in §3.2.1 (main PhD thesis report, p. 42), we only perform online adaptation of the skin colour model; the non-skin model is learnt offline. And in Figure 3.8 (E), we observe cases where parts of the background are mis-labelled as skin. This does not happen with the background of Figure 3.8 (D). Another source of errors is when parts of the hands are not detected as skin due to self-shadowing effects, for example, Figure 3.9 (B).

One aspect where skin model adaptation works well is shown in Figure 3.9 (D) – here we can observe that the skin model is able to adapt to the ethnicity of the signer.



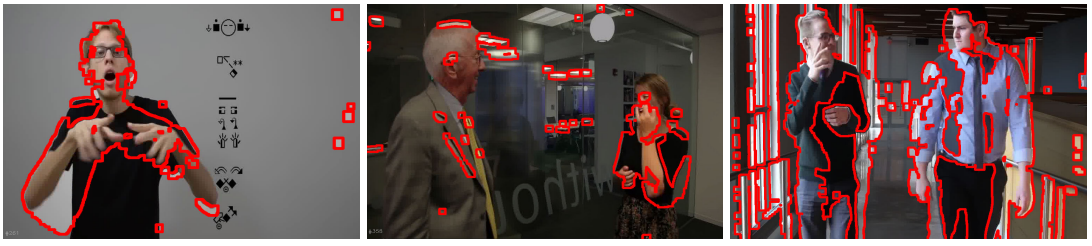
(A) Sample frames from the ECHO NGT dataset. Motion regions are indicated by the red contours.

(B) Sample frames from the BBC Pose dataset, with the difference image shown for the third sample.



(C) Sample frames from the RWTH PHOENIX Weather dataset. The raw difference image and the downsampled and thresholded version of the motion mask are shown for the third sample

(D) Sample frames a "Signing in the Wild" video, exhibiting camera motion and zoom effects. The raw difference image, and thresholded mask are shown for the second sample.



(E) Sample frames from various "Signing in the Wild" videos.

(F) Sample frames for "Signing in the Wild" videos.

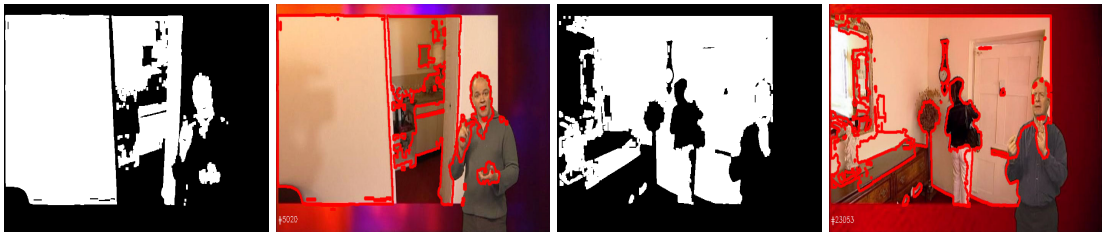
FIGURE 3.7: Motion detection results for various datasets. (Contrast of motion masks has been enhanced for display purposes).

(A) Sample frames with skin results from the ECHO NGT dataset

(B) Correct skin segmentation in the presence of clothes with similar colour

(C) Parts of the signer's clothes are mis-detected as skin due to similarity with skin colour

(D) Correct skin segmentation for BBC Pose sequences when background has skin-like colours. But for the second example, parts of the clothes of the signer are mis-labelled as skin



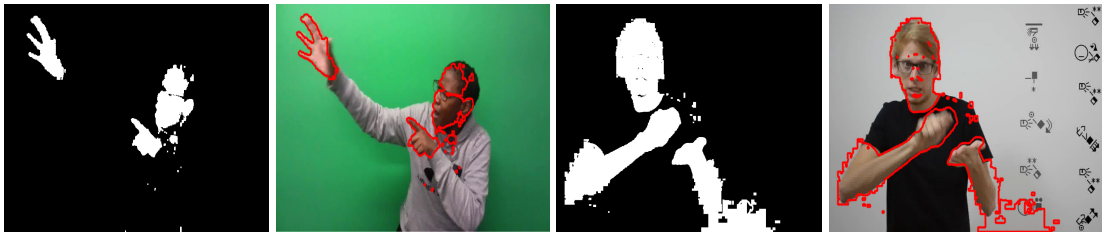
(E) Incorrect skin segmentation caused by background areas being mis-classified as skin

FIGURE 3.8: Skin detection results for the ECHO NGT and BBC Pose datasets

(A) Sample frames from the RWTH PHOENIX dataset with correct skin segmentation

(B) Parts of the hands (indicated by arrows) are not classified as skin. This is due to self-shadowing effects making the colours of parts of the hand deviate from the skin model

(C) Sample frames from the Signing in the Wild dataset



(D) (Left) This example shows successful adaptation of the skin model and skin segmentation to people of different ethnicity. (Right) Successful model adaptation and skin segmentation for a signer wearing glasses.

FIGURE 3.9: Skin detection results for the ECHO NGT and BBC Pose datasets

3.4 KLT feature tracking results

The results provided here complement the ones presented in §B.4 of the main PhD thesis report. In our experiments, we use the KLT version that is provided by the OpenCV library [4] – this consists of the standard KLT algorithm [31–33], with the addition of the pyramidal multi-resolution scheme of Bouguet [42].

Figure 3.10 shows sample frames from the ECHO NGT_AH_fab1 video sequence with superimposed KLT feature tracking results. And Figure 3.11 shows the number of features belonging to the right hand of the signer, and how this varies with time. At points, the number of features being tracked for the signer's hands can get critically low. Another challenge for signing videos is that KLT features tend to have a short lifetime. This is indicated by the histogram in Figure 3.12 below (p. 37). The majority of features are tracked successfully across only 1 frame. The average lifetime of a feature for this particular video sequence is 7.3 frames, while the longest tracked feature had a lifetime of 129 frames.

And Figure 3.13 plots the dissimilarity score ψ of selected features of the same video sequence against time (see main PhD thesis report, item (iii) on page 48). We can observe that the similarity of a feature's current window when compared to the first window of that feature remains approximately stable, with sudden and drastic changes often leading to the loss of the feature. Further investigations show that normally this occurs when the KLT algorithm suffers from a tracking error.

Finally, Figure 3.10 shows sample results for the BBC Pose dataset. When the camera is moving, or there are many skin regions in the FOV, the number of KLT features can go up drastically. We place an upper limit on the maximum number of KLT features that can be tracked at one go.

3.4.1 KLT feature uncertainty

In another set of experiments, we investigate the uncertainty of a KLT feature and how this changes as the feature is tracked from one frame to the next. We represent the initial uncertainty of a newly-selected KLT feature using the method of Kanazawa and Kanatani [46]: that is, we use the inverse of the Hessian matrix as the initial *error covariance* matrix $\mathbf{\Sigma}_u$. We then propagate the feature's uncertainty along its track by incorporating Wong and Majji's method [56], i.e., according to Equation 2.19 given on page 18.

Figures 3.15 and 3.16 show error covariance propagation for two KLT features. The top part of the figures shows the displacements undergone by the KLT feature in the

FIGURE 3.10: KLT feature tracking results for a video of the ECHO NGT dataset. Features are shown as arrows, indicating the displacement from time $t-1$ to current time t. If a KLT feature is deemed to be stationary (not moving), then it is displayed as a red circle.

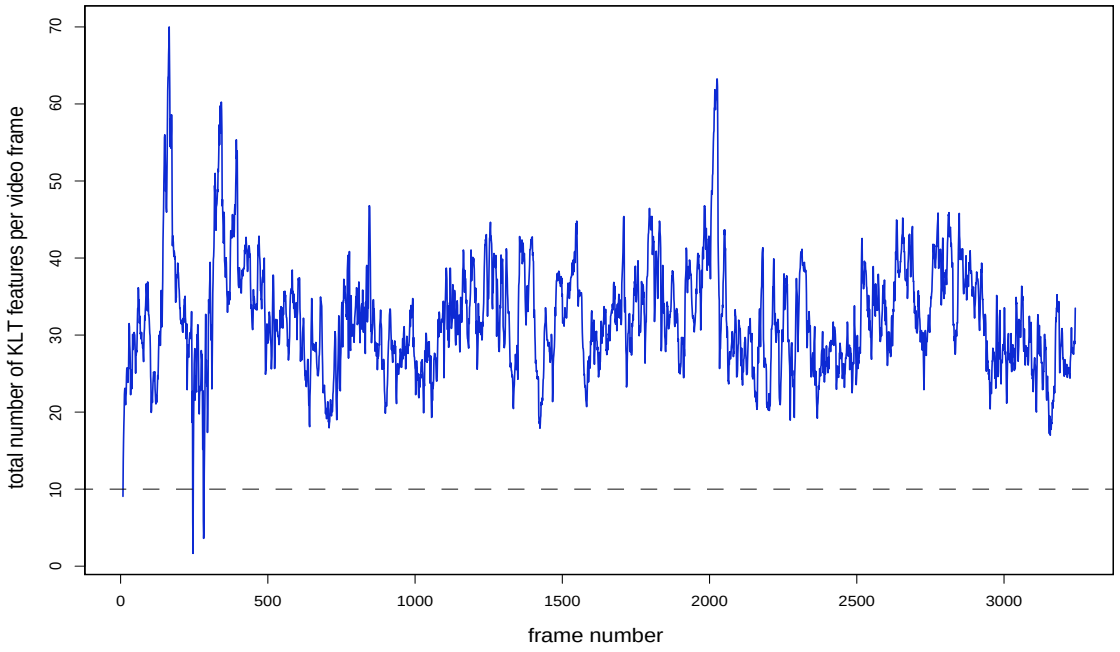


FIGURE 3.11: Number of KLT features for the right hand of the signer (ECHO NGT_AH_fab1 sequence) per video frame. Note that this total drops to below 10 a number of times at the start of the video sequence. Lack of texture, rapid motion and motion blur, contribute to these feature "extinction" events.

image plane (black arrows), together with the error covariances shown as red ellipses. The positions and covariances are also shown on a selection of video frames sampled along the feature's trajectory (centre and lower parts of the figures).

The uncertainty propagation model of the KLT algorithm focuses on its ability to localise a feature given the available gradient (curvature) information in the image data. Depending on how strong the curvature is in the local neighbourhood, and whether this is along a single direction or along two directions perpendicular to each other, will determine by how much the uncertainty of the KLT tracker is increased.

As can be seen from Figures 3.15 and 3.16, this uncertainty model of the KLT algorithm does not capture the full extent of the *true* error for signing videos. In Figure 3.15, there is considerable drift throughout the sequence, especially towards the end (frames #370 – #390). And the amount of drift is not reflected in the size of the error covariances. Effects like aliasing, compression artefacts, image acquisition noise, the effects of the hierarchical pyramids on differentiability and texture, etc., all contribute to the corruption of the gradient information in the image data, and thus add uncertainty to the KLT features. But the main contributing factors for signing videos, appear to be the lack of high-quality texture in the hand areas, as well as the deformable nature of the hands themselves.

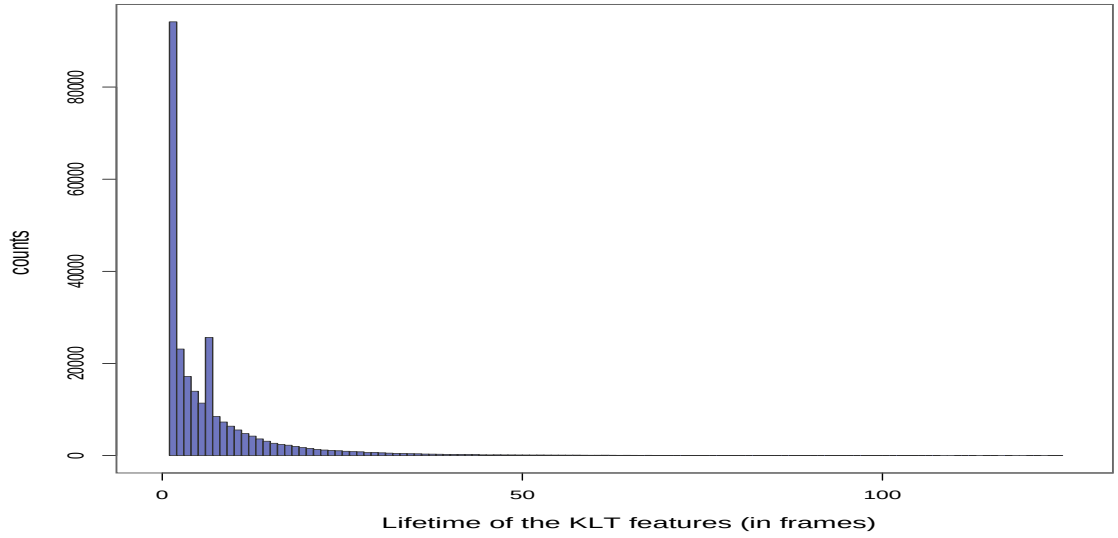


FIGURE 3.12: A histogram of KLT feature lifetimes (in number of video frames) for the ECHO NGT_AH_fab1 video sequence. Note that the absolute majority of KLT features have a very short lifetime.

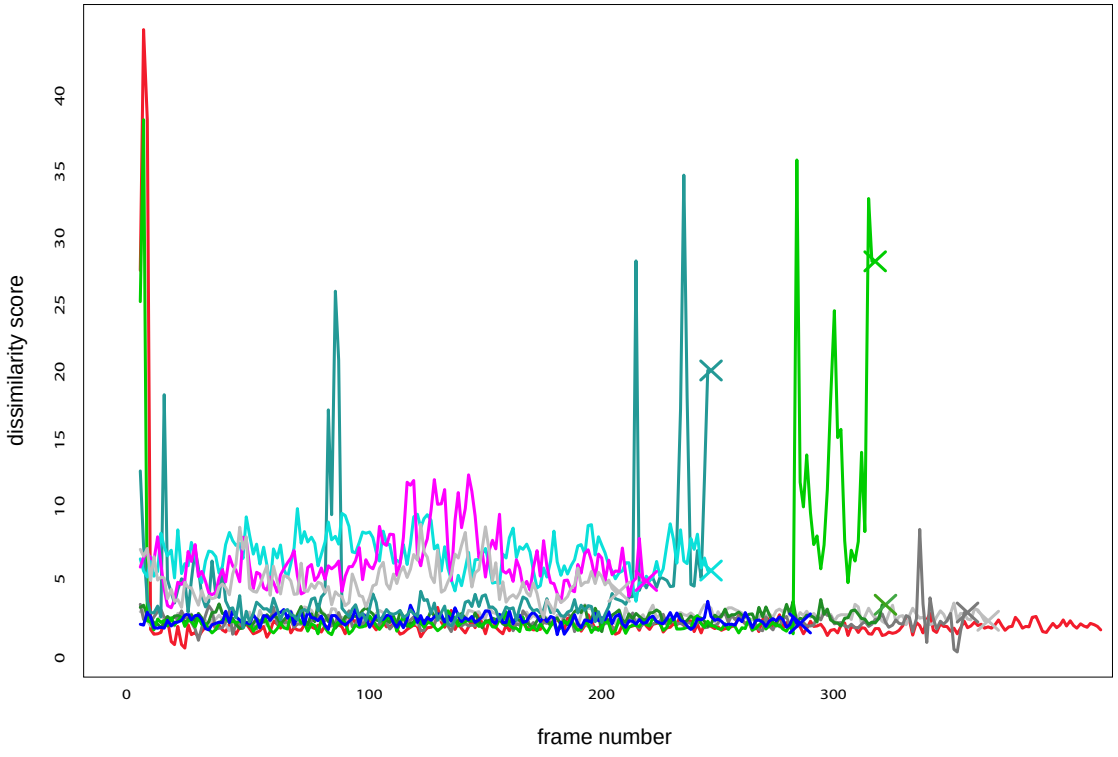


FIGURE 3.13: A plot of dissimilarity score ψ for selected KLT features. This score is based on the sum-of-square difference in appearance between the first window, when a KLT is first created, and the current affine-warped window of the feature. When this difference goes beyond a particular threshold, then the KLT features is marked as 'lost' and deleted (denoted by $\times$ in this figure).

FIGURE 3.14: KLT feature tracking results for the BBC Pose dataset.

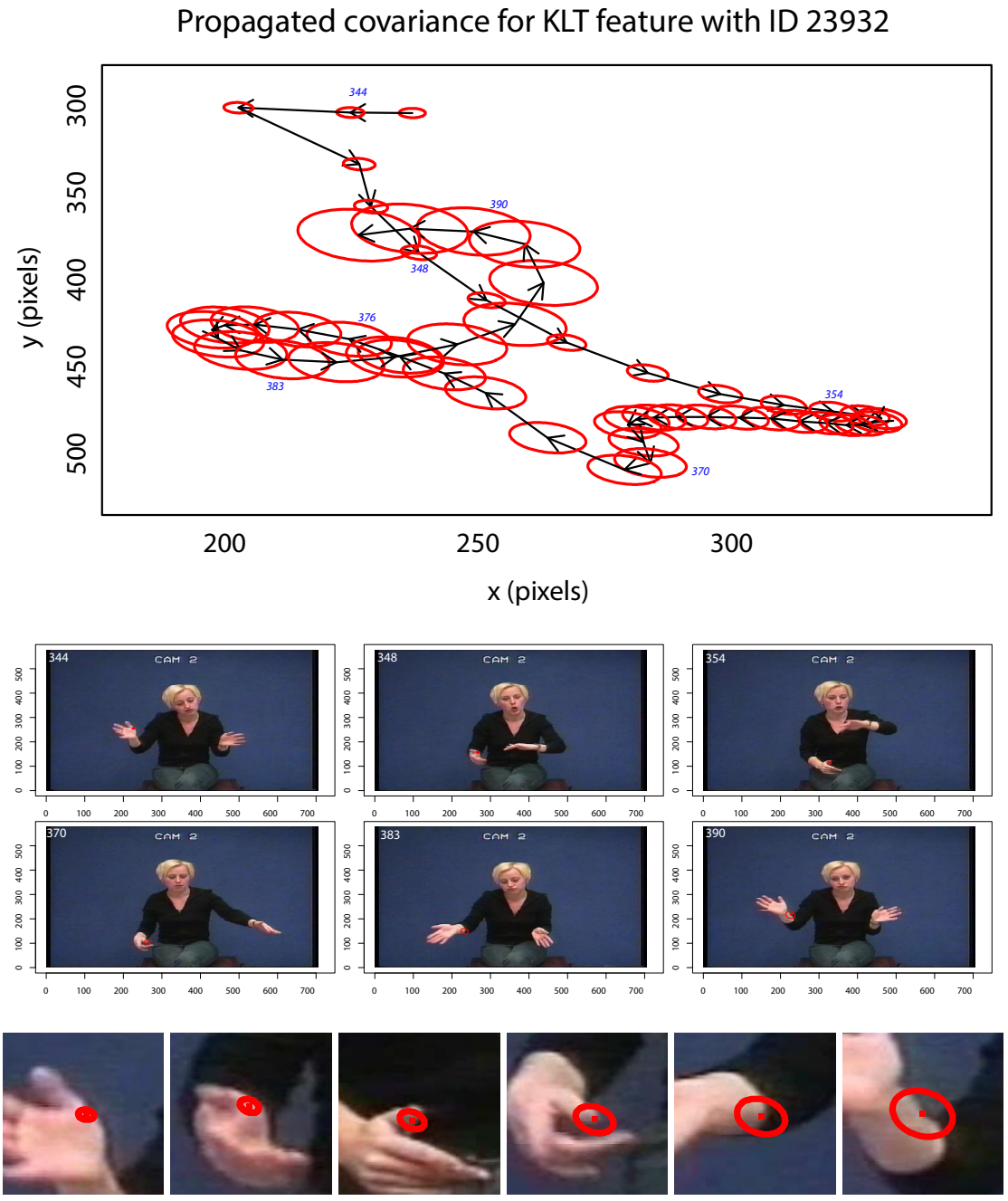


FIGURE 3.15: Propagation of error covariances indicating KLT feature uncertainty. Note the change in position halfway through the feature's lifetime due to a false match caused by hand foreshortening effects – our uncertainty model does not capture feature correspondence errors.

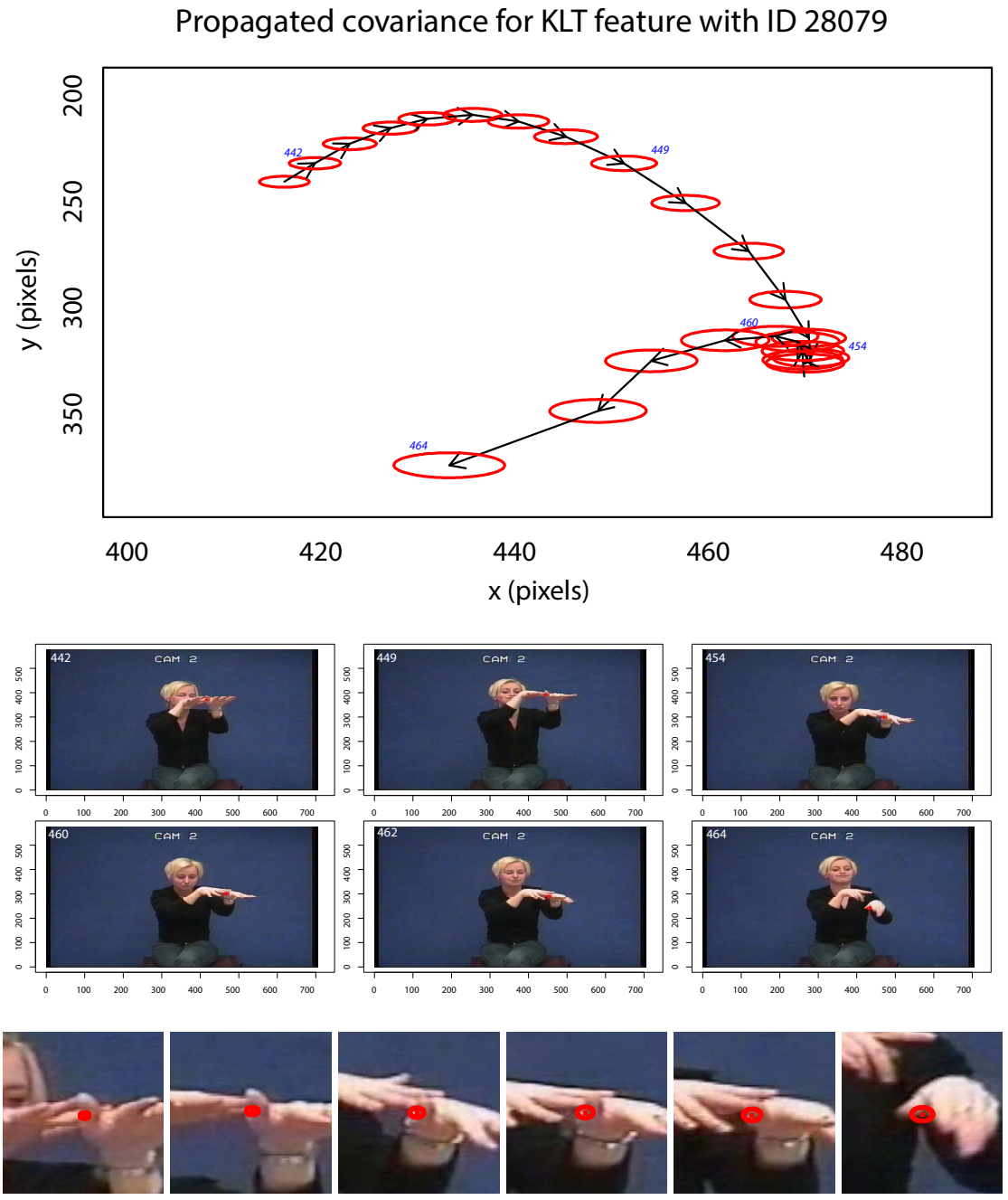


FIGURE 3.16: Propagation of error covariances indicating KLT feature uncertainty – a second example.

3.5 Grouping of features – additional results

This section provides additional results for the experiments conducted in §B.5 of the PhD thesis report. Figures 3.17 to 3.19 show grouping of features for different datasets.

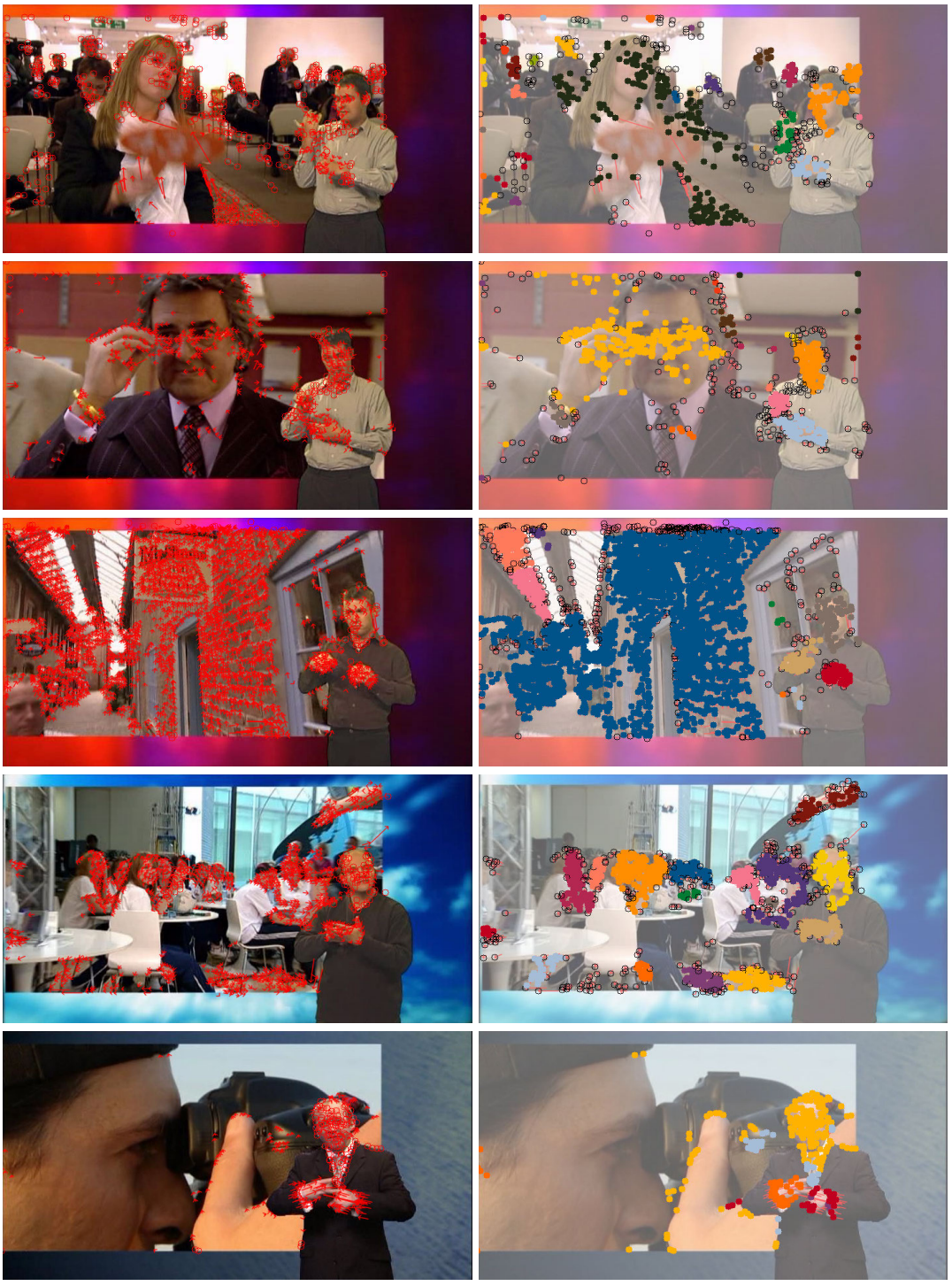


FIGURE 3.17: Grouping of KLT features (left) into hand candidate regions (right) for sample video frames from the BBC Pose dataset. Each cluster is displayed using a different colour. Although here we show all the clusters, in effect, the estimated signing space area will be used to filter many of them out.

FIGURE 3.18: Grouping of features into hand candidate regions for sample video frames from the 'Signing in the Wild' dataset. Each cluster of features is displayed using a different colour.

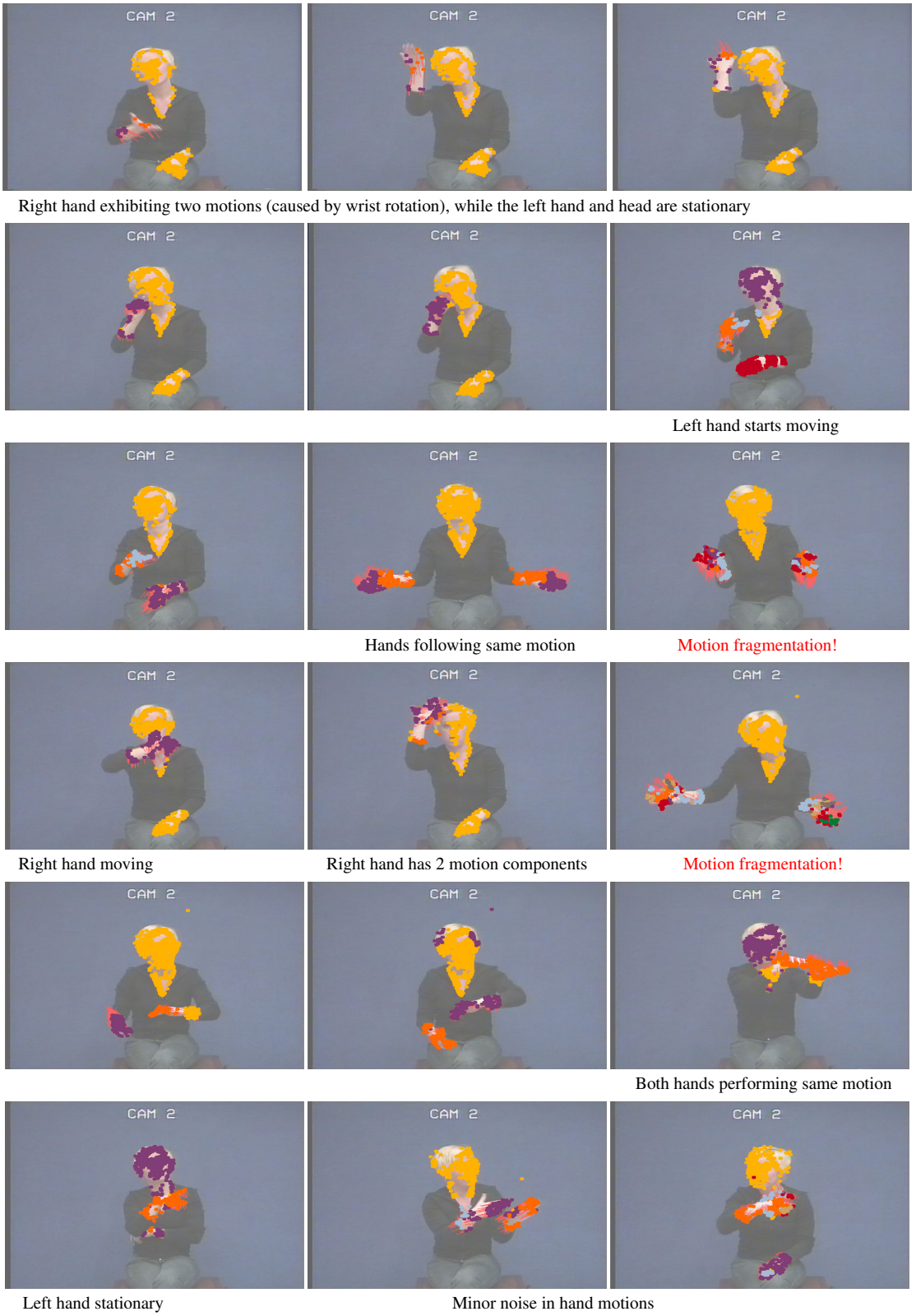


FIGURE 3.19: Grouping of features into hand candidate regions. Each cluster is indicated by a different colour. Note that a cluster's colour is not maintained over time, so a cluser in the current frame can be shown in a different colour in the next frame.

3.6 Long-term hand tracking – additional results

This section contains additional results on long-term hand tracking, generated as part of the experiments of §B.6 of the main PhD thesis report. These results are shown in Figures 3.20 to 3.22 below.

In these figures, the tracker's positions of the left and right hand are shown as bright red and bright green circles respectively (solid circle if the tracker associated an observation with the hand target in the current frame; open circle if the hand has not been seen in the current frame). The observations are shown as open black circles – an observation is a cluster of KLT features, with the features grouped together because they all fit a particular motion model. We only show the centroids of the clusters (observations) and the hands in these figures. Also displayed are the individual features coloured by the group they belong to (shown in darker colours), as well as the signer's face detections (enclosed in a white bounding box).

FIGURE 3.20: Successful tracking of the right hand (green solid circle) through clutter (multiple observations due to motion fragmentation when grouping features; shown as black open circles). Note especially the increase in clutter starting from the third frame in the sequence. Final frame shows a superimposition of the hand trajectories.

FIGURE 3.21: Another example of successful tracking of the signer's hands.

FIGURE 3.22: Tracking results for the BBC pose sequence

Chapter 4

Signing Detection system

Figure 4.1 illustrates our complete signing detection system of §4 of the main PhD thesis report. This system consists of a two-stream CNN and RNN network. The RNNs are unrolled in time, to better illustrate how the CNN features from the image and motion streams are passed as input sequences to the RNNs.

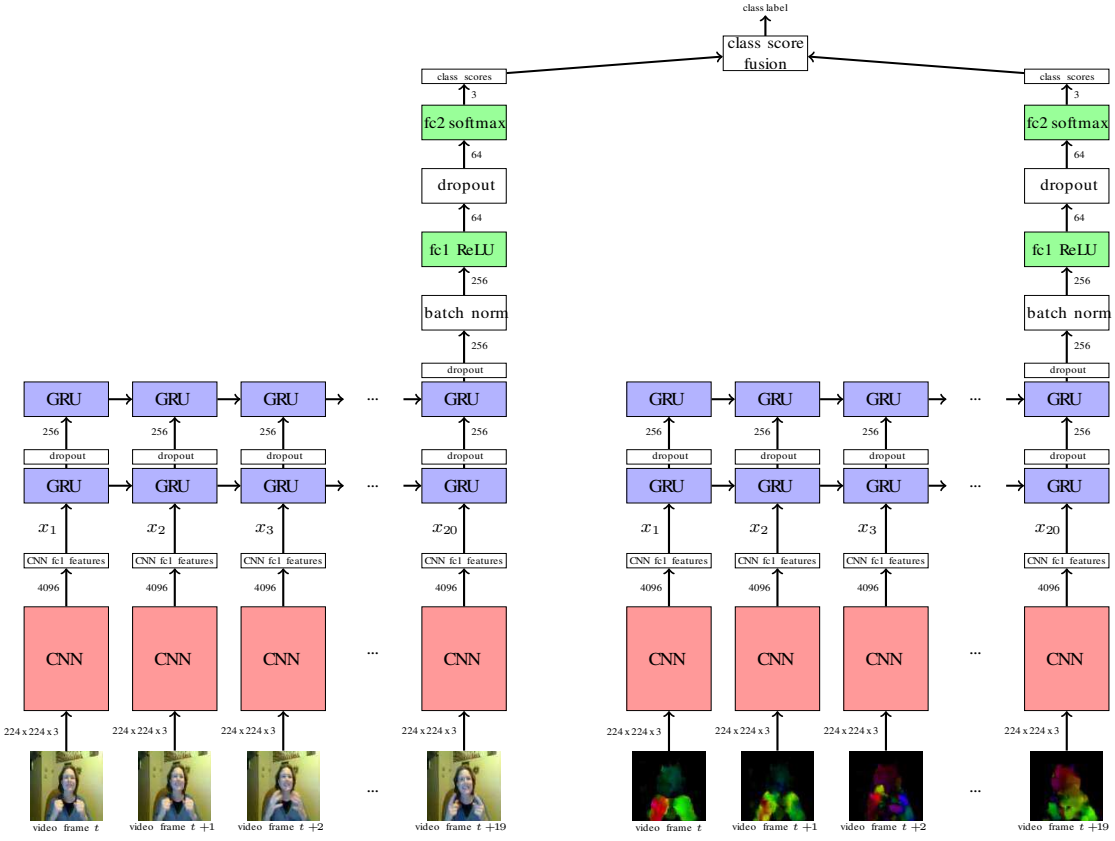


FIGURE 4.1: Our proposed signing detection system incorporating a 2-stream CNN network, one stream processing image data, the other motion data (optical flow in this case). A block of 20 timesteps from each CNN are fed into a 2-layer RNN, which in turn is fed into a non-linear classifier. Finally, class score fusion combines the predictions from both classifiers.

Chapter 5

The Kalman filter

The Kalman filter is a *recursive* least squares estimator, making it ideal for use in an online setting, i.e., when new information becomes available in a sequential (incremental) manner. At each time step, the filter incorporates this new information to update and improve its estimate. Furthermore, the Kalman filter adopts a probabilistic approach – both the estimate and the new information have an associated uncertainty. The Kalman filter combines the two together via a *weighted* approach, whereby each piece of information is weighted by its measure of certainty [88].

Considering a simple case, let x_1 be the filter's estimate, and let x_2 denote an incoming measurement or observation. The difference between the two $(x_2 - x_1)$ represents the new information, or *innovation*.

Both x_1 and x_2 have an associated uncertainty given by σ_1 and σ_2 respectively, assuming that a Gaussian can be used to characterise the values and their uncertainties. We define the filter's *update gain* as:

$$K = \frac{(\sigma_1)^2}{(\sigma_1)^2 + (\sigma_2)^2} \tag{5.1}$$

And we combine the estimate and the measurement to get a new improved estimate x_3 as follows:

$$x_3 = x_1 + K(x_2 - x_1) \tag{5.2}$$

The uncertainty σ_3 of the new estimate x_3 is given by:

$$(\sigma_3)^2 = (1 - K)\,(\sigma_1)^2 \tag{5.3}$$

Equation 5.2 is essentially a weighted sum of the previous estimate x_1 and the new

information (the innovation). Weighting is given by the update gain K, which is determined by the relative uncertainties of the two values in Equation 5.1. Thus the Kalman filter essentially performs *information fusion* of the "old information" x_1 and the "new information" $(x_2 - x_1)$, in order to arrive at a better estimate x_3. The recursive nature of the Kalman filter is evident from the iterative forms of Equations 5.2 and 5.3.

In the above simplified case, x_1 and x_2 are assumed to have the same informational representation. Now we extend the formulation of the Kalman filter by assuming an underlying *state space model* that describes how the object being estimated behaves dynamically [88]. The filter's estimate of the state at time t is now represented by vector $\hat{\mathbf{x}}_t$ (with 'hat' symbol) belonging to some *state space*, which is different from the *measurement space* to which a measurement $\mathbf{z}_t$ at time t belongs to. Typically the measurement space is a subspace of the state space, with some elements of the object's state being unobserved.

And prior to combining the old information with the new information in Equation 5.2, a *prediction phase* is applied, so that the model can be "fast-forwarded" to the time instant just before we add the new information. Thus the Kalman filter combines the *predicted* information at time t with the new information (the innovation) at time t.

The predicted state (the *a priori* state estimate) just prior to the update with the new information is typically denoted by $\hat{\mathbf{x}}_t^-$ (with superscript '$-$' symbol), and matrix $\mathbf{P}_t^-$ being the corresponding state error covariance estimate. Similarly, we denote by $\hat{\mathbf{x}}_t^+$ (with superscript '$+$' symbol) and $\mathbf{P}_t^+$, the *a posteriori* state estimate and corresponding error covariance after the update is done [89, pp.348–358].

Relating the state space with the measurement space, we have the following *measurement equation*:

$$\mathbf{z}_t = \mathbf{H}\,\hat{\mathbf{x}}_t + v_t \tag{5.4}$$

where $\mathbf{z}_t$ is a measurement at time t, $\hat{\mathbf{x}}_t$ is the estimated state at time t, and matrix $\mathbf{H}$ defines the mapping from the state space to the measurement space. The measurement noise v_t is a Gaussian random variable with zero mean and covariance $\mathbf{R}$, so its probability distribution is given by:

$$p(v) \approx \mathcal{N}(0, \mathbf{R}) \tag{5.5}$$

The predicted state at time t is given by the following equations:

$$\hat{\mathbf{x}}_t^- = \mathbf{A}\hat{\mathbf{x}}_{t-1}^+ + \mathbf{B}u_{t-1} \tag{5.6}$$

$$\mathbf{P}_t^- = \mathbf{A}\,\mathbf{P}_{t-1}^+\,\mathbf{A}^T + \mathbf{Q} \tag{5.7}$$

where $\mathbf{A}$ is the state transition matrix, $\mathbf{P}$ is the state covariance error, $\mathbf{B}$ is the input or control matrix, u is the input or control vector, and $\mathbf{Q}$ is the process noise covariance matrix [88].

Then given a measurement $\mathbf{z}_t$ at time t, the innovation q_t and innovation covariance matrix $\mathbf{S}_t$ are given by:

$$q_t = \mathbf{z}_t - \mathbf{H}\,\hat{\mathbf{x}}_t^- \tag{5.8}$$

$$\mathbf{S}_t = \mathbf{H}\,\mathbf{P}_t^-\,\mathbf{H}^T + \mathbf{R} \tag{5.9}$$

Finally, the weighted update of the prediction with the new measurement is computed via the following equations, with matrix $\mathbf{K}_t$ being the Kalman gain at time t:

$$\mathbf{K}_t = \mathbf{P}_t^-\,\mathbf{H}^T\left(\mathbf{H}\,\mathbf{P}_t^-\,\mathbf{H}^T + \mathbf{R}\right)^{-1} = \mathbf{P}_t^-\,\mathbf{H}^T\,\mathbf{S}_t^{-1} \tag{5.10}$$

$$\hat{\mathbf{x}}_t^+ = \hat{\mathbf{x}}_t^- + \mathbf{K}_t\,q_t \tag{5.11}$$

$$\mathbf{P}_t^+ = (\mathbf{I} - \mathbf{K}_t\,\mathbf{H})\,\mathbf{P}_t^- \tag{5.12}$$

Figure 5.1 summarises the Kalman filter formulation covered in this section [88, 89]. Note that for the 1D case, and setting the state space to be the same as the measurement space, we get: $\mathbf{H} = 1$, $\mathbf{A} = 1$, $\mathbf{P}_t^+ = \hat{\sigma}_t^2$, and $\mathbf{P}_t^- = \hat{\sigma}_{t-1}^2$. Thus Equation 5.10 simplifies to Equation 5.1, Equation 5.11 to 5.2, while Equation 5.12 becomes 5.3.

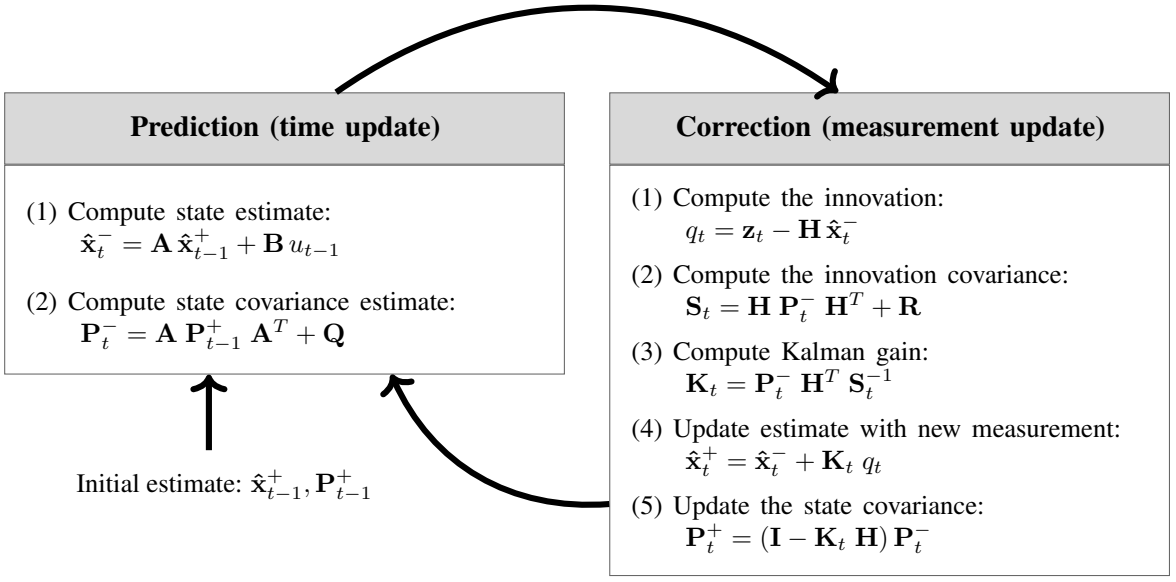


FIGURE 5.1: Kalman Filter formulation

5.1 Kalman filter consistency

One way of checking the Kalman filter for consistency is to check whether the innovation q_t of Equation 5.8 is bounded by:

$$\pm \kappa \sqrt{\mathbf{S}_t} \tag{5.13}$$

for some pre-defined value of κ, typically set to 2 or 2.5. This test is referred to as the *innovation magnitude bounds test* in Gamse et al. [90].

In our experiments, we utilise the innovation magnitude bounds test as a simple outlier rejection scheme. In this scheme, outliers are added to the measurement Equation 5.4 as given below:

$$\mathbf{z}_t = \mathbf{H}\,\hat{\mathbf{x}}_t + v_t + \Upsilon_t\, m_t \tag{5.14}$$

where Υ_t gives the probability of outlier appearance and m_t is the magnitude of the outlier [86]. In this outlier rejection scheme, we reject measurements with $\Upsilon_t = 1$, and accept measurements with $\Upsilon_t = 0$. Detection of hypothesis $\Upsilon_t = 0$ is based on the observation that the innovation q_t is Gaussian distributed with zero mean and covariance $\mathbf{S}_t$. The hypothesis condition is then given by:

$$|q_t| \;\leq\; \kappa \sqrt{\mathbf{S}_t} \tag{5.15}$$

Section 3.2 describes one such usage for tracking the face of the signer, and Figure 3.5 illustrates the innovation bounds and rejected outlier measurements.

Chapter 6

Multiple Hypothesis Tracking (MHT)

This chapter provides additional information related to the MHT algorithm, including the full derivation of the hypothesis likelihood equation given by Equation 5.4 (main PhD thesis report, §5.3.1, p. 104).

6.1 Multiple Object Tracking (MOT) formulation

In a multiple object tracking (MOT) situation, we are interested in tracking a set of moving targets, where each target forms a trajectory over time. Each target in this set has a time of birth (when first acquired), an evolution in *state space* (describing its motion and behaviour), and potentially a time of death (track termination; when the target is no longer visible).

The set of N targets maintained by an MOT tracker at time t is denoted by $\mathbf{X}(t) = \{\mathbf{x}_k(t)\}_{k=1}^{N}$, where each $\mathbf{x}_k(t)$ represents the state of a target k at time t. We assume that a target state depends on the previous states via the *state evolution equation*:

$$\mathbf{x}_k(t) = f_t(\,\{\mathbf{x}_k(t-1)\,, \mathbf{x}_k(t-2)\,, \ldots, \mathbf{x}_k(1)\}\,) + \mathbf{w}(t) \tag{6.1}$$

where $\mathbf{w}(t)$ in the above equation represents a random noise component, while function $f_t(\cdot)$ describes the deterministic part in the evolution from a target's previous states to the current state $\mathbf{x}_k(t)$. Function f_t can potentially be time-varying. We denote the *state history* of a particular target k by $\mathbf{x}_k^t = \{\mathbf{x}_k(t)\,, \mathbf{x}_k(t-1)\,, \ldots, \mathbf{x}_k(1)\} = \{\mathbf{x}_k(i)\}_{i=1}^{t}$, and denote the set of all state histories of all targets (i.e., tracks) as:

$$\mathbf{X}^t = \bigcup_{i=1}^{t} \mathbf{X}(i)\,. \tag{6.2}$$

In a MOT situation, the total number of targets is unknown and the number of active targets at any moment in time is variable. Targets can have different *temporal support*, i.e., lifetimes of targets k and l, with states $\mathbf{x}_k$ and $\mathbf{x}_l$ respectively, may not cover the same temporal range. This makes MOT a much harder problem to solve [91].

During the tracking process, a number of measurements (observations) are acquired by the sensor, where some (but not all) of the measurements originate from the targets of interest. We denote the set of measurements observed at time t as $\mathbf{Z}(t) = \{\, \mathbf{z}_j(t) \,\}_{j=1}^{M_t}$, with M_t being the total number of measurements observed at time t. Note that M_t varies in time. And we denote the set of all measurements for all time steps from 1 to t as:

$$\mathbf{Z}^t = \bigcup_{i=1}^{t} \mathbf{Z}(i) \tag{6.3}$$

If a current measurement $\mathbf{z}_j(t)$ is determined to originate from a specific target k, then the target's state $\mathbf{x}_k(t)$ can be updated to incorporate the new information provided by the measurement $\mathbf{z}_j(t)$. We assume that this relationship can be expressed via the following *measurement equation*:

$$\mathbf{z}_j(t) = h_t(\mathbf{x}_k) + \mathbf{v}(t) \tag{6.4}$$

where function $h_t(\cdot)$ links the measured features with the state variables, while $\mathbf{v}(t)$ represents random noise in the measurement process. Many techniques are available for estimating the target state $\mathbf{x}_k$ from its associated set of measurements. Commonly used techniques for kinematic state estimation include the Kalman filter and its extensions, sequential Monte Carlo (SMC) methods [92], and square root information filtering (SRIF) techniques [93].

Given a set of measurements $\mathbf{Z}^t$, an MOT tracker needs to determine the most likely set of tracks $\mathbf{X}^t$ that can best explain the observed data. In other words, an MOT tracker estimates:

$$P\big(\mathbf{X}^t \,|\, \mathbf{Z}^t\big) \tag{6.5}$$

As alluded to earlier, the unknown number of targets, variations in the temporal support of each target (different lifetimes), the problem of missing measurements (targets sometimes unobserved), and false alarms, all conspire to make solving $P(\mathbf{X}^t \,|\, \mathbf{Z}^t)$ very challenging. In addition, Coraluppi [91] state that directly estimating Equation 6.5 above is "conceptually difficult" from a mathematical estimation point-of-view.

6.2 The MHT estimation approach

Rather than solving Equation 6.5 directly, the MHT algorithm treats the data association problem separately from the target state estimation problem. The data association problem is tackled using a MAP estimate via the use of Bayesian inference, while the target state is estimated using a minimum mean square error (MMSE) estimator, conditioned on the association estimate [91, 94]. This is described by the following equations:

$$\hat{\Omega}^t_{\text{MAP}} = \arg\max_{\Omega^t_i} P\left(\Omega^t_i \mid \mathbf{Z}^t\right) \tag{6.6}$$

$$\hat{\mathbf{X}}^t_{\text{MMSE}} = \arg\max_{\mathbf{X}^t} P\left(\mathbf{X}^t \mid \mathbf{Z}^t, \hat{\Omega}^t_{\text{MAP}}\right) \tag{6.7}$$

where $\{\Omega^t_i\}$ in Equation 6.6 represents the set of all potential solutions (hypotheses) to the data association problem and $\hat{\Omega}^t_{\text{MAP}}$ represents the MAP estimate from this set. Then given $\hat{\Omega}^t_{\text{MAP}}$, the target states can be estimated via Equation 6.7. The MHT algorithm employs an estimator like the Kalman filter for performing the target kinematic state estimate (based on Equations 6.1 and 6.4). Note that in effect, the MHT algorithm integrates a multiple hypothesis data association (MHDA) component (Equation 6.6) with multiple independent track filters for kinematic state estimation (Equation 6.7).

For each possible data association solution Ω^t_i in Equation 6.6, the MHT algorithm considers *explicit* explanations for the observed data. In other words, in a given time step, specific measurements get associated with specific targets, other measurements are labelled as false alarms (clutter) or as new targets, and targets with unassociated measurements are labelled as either undetected or dead. To simplify the data association problem, the MHT algorithm assumes that the so-called *uniqueness constraint* holds, i.e., that a target can give rise to at most one measurement.

What differentiates the MHT algorithm from other MOT methods (like the GNN or JPDA filter [92]), is that instead of just considering the most likely (or a single) hypothesis only, MHT keeps multiple data association hypotheses Ω^t_i at any one time. And using its *deferred logic* property, it can eventually make a more informed decision as to which is the most likely data association solution. As more evidence is accrued over time, eventually one (the correct) hypothesis dominates in a statistical sense over the others.

Figure 6.1 shows the architecture of the MHT algorithm, with its major conceptual components, including the use of the mixed MAP-MMSE estimation approach for data association hypothesis evaluation. And Figures 6.2 and 6.3 illustrate, via an example,

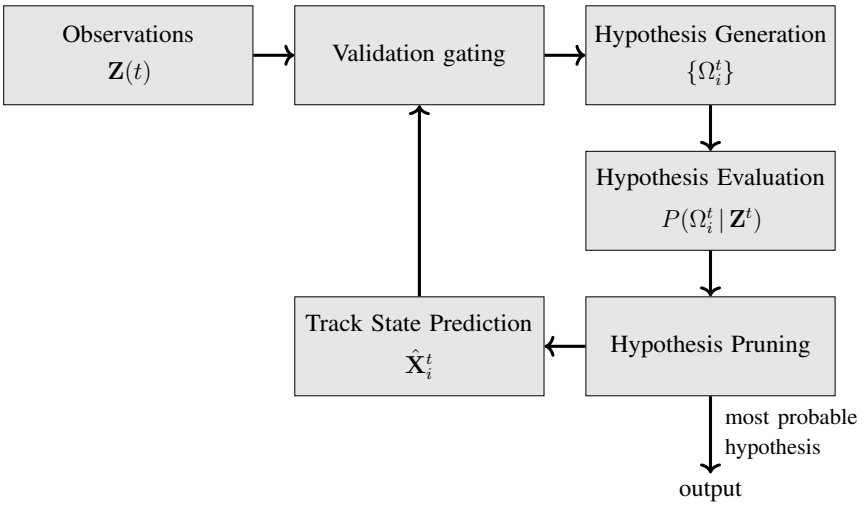


FIGURE 6.1: The MHT architecture (Adapted from: Amditis et al. [95])

the multiple hypothesis generation method of MHT, highlighting the way it uses explicit explanations to interpret the observed data.

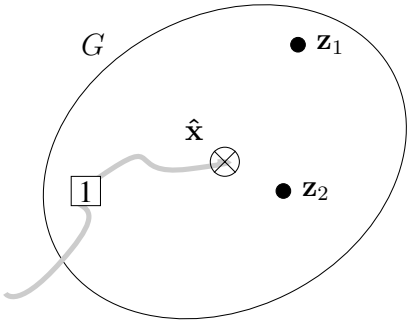
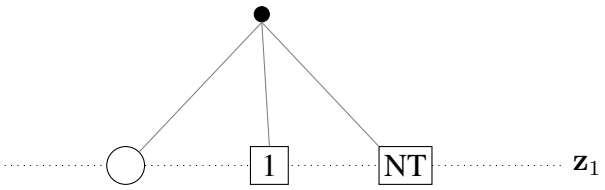


FIGURE 6.2: An example of a target with its trajectory shown in gray. Its predicted position $\hat{\mathbf{x}}$ at time t is denoted by $\otimes$ and the validation gate region G is depicted as an ellipse. Two new measurements $\mathbf{z}_1$ and $\mathbf{z}_2$ are observed within the object's validation gate at time t. Various interpretations can be used to describe the current situation, thus giving rise to multiple association hypotheses (see Figure 6.3).

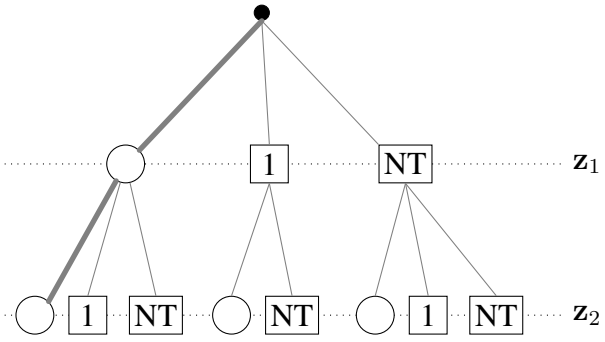
6.3 MHT hypothesis likelihood formulation

At each timestep in the tracking process, multiple interpretations are possible when associating measurements with targets. Let us denote one such explanation (hypothesis) as Ω_i^t; we index this explanation (one of many possible explanations up to time t) by i. This hypothesis associates the set of measurements $\mathbf{Z}^t$ up to time t with existing targets, new targets, or clutter. And the set of all potential hypotheses up to time t is denoted by:

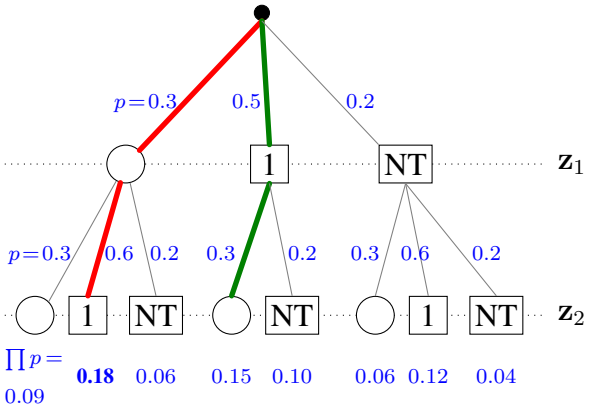
$$\Omega^t = \left\{\Omega_i^t\right\}_{i=1}^{H_t} \tag{6.8}$$



(A) This figure illustrates the process of generating association hypotheses for the tracking example given in Figure 6.2. The hypotheses are best represented in terms of a *hypothesis tree*. The root of the tree (denoted by a black circle) represents our knowledge as of time $t-1$: this consists of existing target 1 with predicted position $\hat{\mathbf{x}}$. Given new measurements observed at time t, we go through each measurement, adding new hypotheses to the tree. Measurement $\mathbf{z}_1$ can either be a false alarm, a re-detection of target 1 at time t, or caused by a new target. We represent these 3 possibilities by adding 3 new branches to the hypothesis tree: we use the symbols ◯, 1 , and NT for the 3 cases respectively.



(B) For measurement $\mathbf{z}_2$, we add new tree branches for each possible interpretation that satisfies the uniqueness constraint of MHT. There are now 8 possible hypotheses for both measurements: the leftmost tree branch (in gray) represents the hypothesis that both $\mathbf{z}_1$ and $\mathbf{z}_2$ are false alarms; the next hypothesis stands for $\mathbf{z}_1$ being a false alarm, and $\mathbf{z}_2$ being a re-detection of target 1, etc.



(C) One can surmise that not all hypotheses are equally probable. In this example it is reasonable to assume that hypothesis #2 (red) is more probable than hypothesis #4 (green), simply because the distance (innovation) between the predicted position of track 1 and $\mathbf{z}_2$ is less than that to $\mathbf{z}_1$. But rather than being forced to decide now (with the limited data available), we can continue adding new branches to the tree in successive timesteps and observe how the hypothesis probabilities (examples in blue) evolve with time. With more data, we can arrive at a more informed and thus better decision.

FIGURE 6.3: Multiple hypothesis generation example

where H_t is the number of all hypotheses generated at time t.

The MHT algorithm adopts a recursive Bayesian approach for hypothesis generation. The recursive formulation is defined as follows: the set of current hypotheses Ω^t is formed from the set of *prior hypotheses* Ω^{t-1} at time $t-1$ (also called the *parent hypothesis set*), and the set $\psi(t)$ of possible associations linking the current observations $\mathbf{Z}(t)$ with the targets $\mathbf{X}^{t-1}$. For a particular hypothesis Ω_i^t, the assignment set $\psi_i(t)$ associates each measurement $\mathbf{z}_j(t)$ in the current time step either to an existing track, a false alarm, or a new track. Note that the set $\psi(t)$ constitute the new branches added to the hypothesis tree in the current time step. Thus a particular hypothesis Ω_i^t is defined recursively as [96–98]:

$$\Omega_i^t = \left\{ \psi_i(t) \, , \Omega_{g(i)}^{t-1} \right\} \tag{6.9}$$

where the function $g(i)$ gives the index of the parent hypothesis. In relation to the parent (prior) hypothesis, the set of hypotheses Ω^t can be referred to as the *child hypotheses* or *posterior hypotheses*.

Figure 6.4 illustrates this recursive formulation defined by Equation 6.9. Note that

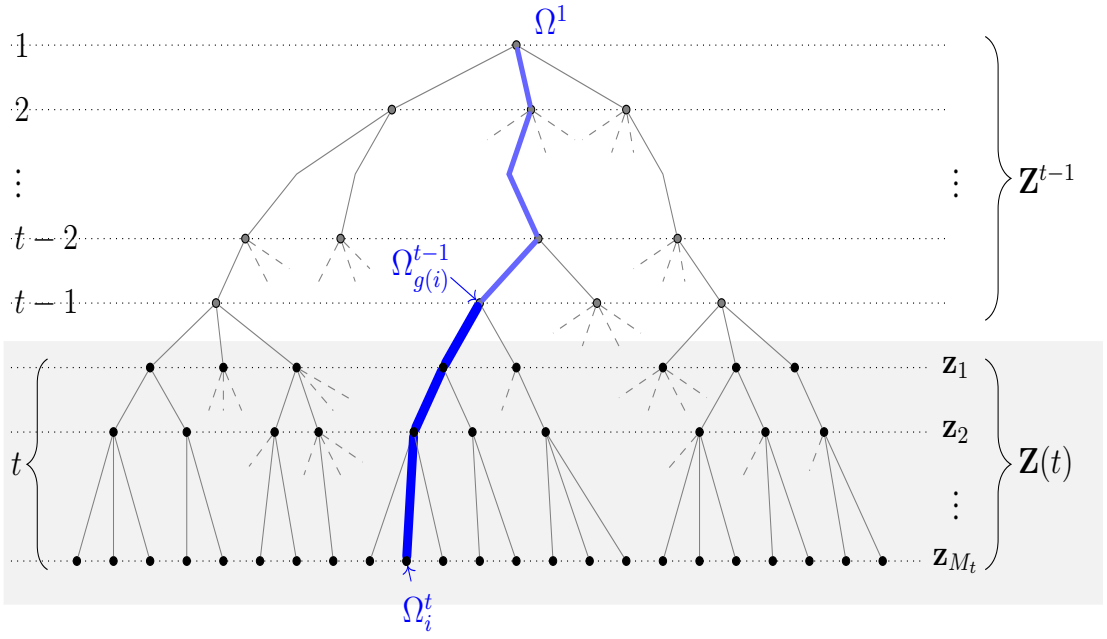


FIGURE 6.4: Recursive hypothesis formulation. Each path of the tree from the root node to a leaf node denotes a particular tracking hypothesis, e.g., Ω_i^t highlighted in blue. Ω_i^t is a child hypothesis of the parent hypothesis $\Omega_{g(i)}^{t-1}$, which in turn is a child of one of the hypothesis in Ω^{t-2}, and so forth until the root node Ω^1 of the hypothesis tree is reached. In each time step, the tree grows by M_t levels, one level for each measurement. The new branches that link the child hypothesis Ω_i^t to its parent $\Omega_{g(i)}^{t-1}$ define the data association set $\psi_i(t)$ (thicker blue lines).

the hypothesis tree grows by M_t levels in each time step, corresponding to the M_t measurements detected at time t. The set $\psi_i(t)$ of a hypothesis Ω_i^t consists of the new branches joining it to its parent hypothesis [1]. An alternative way to represent $\psi_i(t)$ is as a similarity matrix (a *hypothesis matrix* [80]), with measurements on one side and targets on the other.

We will now derive the probability of hypothesis Ω_i^t needed by the MHT algorithm in order to estimate Equation 6.6. We have:

$$
\begin{aligned}
& P\left(\Omega_i^t \mid Z^t\right) \\
&= P\left(\psi_i(t)\,, \Omega_{g(i)}^{t-1} \mid Z^t\right) && \text{(...via the recursive formulation of } \Omega_i^t \text{ as given by Equation 6.9)} \\
&= P\left(\psi_i(t)\,, \Omega_{g(i)}^{t-1} \mid Z(t)\,, Z^{t-1}\right) \\
&= \frac{P\left(\psi_i(t)\,, \Omega_{g(i)}^{t-1}, Z(t)\,, Z^{t-1}\right)}{P(Z(t)\,, Z^{t-1})} && \text{(...via conditional probability definition)} \\
&= \frac{P\left(Z(t) \mid \psi_i(t)\,, \Omega_{g(i)}^{t-1}, Z^{t-1}\right)\; P\left(\psi_i(t) \mid \Omega_{g(i)}^{t-1}, Z^{t-1}\right)\; P\left(\Omega_{g(i)}^{t-1} \mid Z^{t-1}\right)\; P(Z^{t-1})}{P(Z(t)\,, Z^{t-1})} \\
& && \text{(...via conditional probability chain rule)} \\
&= \frac{P(Z^{t-1})}{P(Z(t), Z^{t-1})}\, P\left(Z(t) \mid \psi_i(t)\,, \Omega_{g(i)}^{t-1}, Z^{t-1}\right) P\left(\psi_i(t) \mid \Omega_{g(i)}^{t-1}, Z^{t-1}\right) P\left(\Omega_{g(i)}^{t-1} \mid Z^{t-1}\right) \\
& && \text{(...after re-arranging the order of terms)} \\
&= \overbrace{\frac{1}{c}}^{\text{evidence}}\; \underbrace{P\left(Z(t) \mid \psi_i(t)\,, \Omega_{g(i)}^{t-1}, Z^{t-1}\right)}_{\text{factor 1}}\; \overbrace{\underbrace{P\left(\psi_i(t) \mid \Omega_{g(i)}^{t-1}, Z^{t-1}\right)}_{\text{factor 2}}}^{\text{likelihood}}\; \overbrace{\underbrace{P\left(\Omega_{g(i)}^{t-1} \mid Z^{t-1}\right)}_{\text{factor 3}}}^{\text{prior}}
\end{aligned}
\tag{6.10}
$$

In Equation 6.10 above, the normalisation factor c is set to:

$$
c = \frac{P(Z(t)\,, Z^{t-1})}{P(Z^{t-1})} = P\left(Z(t) \mid Z^{t-1}\right) \tag{6.11}
$$

and it can be easily seen that c is independent of any particular hypothesis and is constant for any Ω_i^t [96]. The definition of conditional probability and the chain rule used in

[1] One may raise the concern that by adding the M_t levels to the hypothesis tree in a particular order, probably based on some arbitrary measurement labelling order during acquisition, we might be forcing an *a priori order* on the data and thus introducing some form of statistical bias in the MHT algorithm. Actually, such a bias effect does not exist in the MHT algorithm [99]. But one has to keep this in mind if pruning techniques are applied to the hypothesis tree, so as not to inadvertently introduce such biases.

Equation 6.10 are reproduced in a footnote below [2] for convenience.

We can observe that Equation 6.10 is recursive in nature, since the probability of a hypothesis is defined in terms of its parent hypothesis. Additionally we can observe that Equation 6.10 is composed of the probability of the parent hypothesis (the *prior*), terms that give the *likelihood* of the measurements and the new data association hypothesis at time t, and a final term that gives the probability of the *evidence* of the data. Thus, we can conclude that the MHT algorithm follows the principle of Bayesian inference when it comes to estimating $P(\Omega_i^t \mid Z^t)$.

Equation 6.10 embodies the Bayesian evidence accrual characteristic of the MHT algorithm, recursively evaluating the probability on multiple association hypotheses and enable the correct hypothesis to dominate competing hypotheses over time [93]. It provides a straightforward and theoretically sound method of recursively updating belief about unknown random variables in a MOT scenario by incorporating new information over time [93].

We will now go through the three main factors of Equation 6.10 and derive each of the individual probability terms.

Factor 3 of Equation 6.10 is the probability of the parent hypothesis and defines the recursive nature of the MHT probability equation.

Factor 1 of Equation 6.10 gives the likelihood of the current measurements $Z(t)$ given the data association hypothesis at time step t and the parent hypothesis. We saw earlier that a measurement can be (i) associated with an existing target (a true detection), (ii) the measurement can be caused by the birth of a new target, or (iii) it could be a false alarm.

For a measurement $\mathbf{z}_j$ associated with a specific target $\mathbf{x}_k$, the probability of the association can be given by how "similar" the measurement is to the predicted state $\hat{\mathbf{x}}_k$ of the target at time t. Since the MHT algorithm employs an estimator like the Kalman filter, we can use the *innovation* returned by the estimator as the similarity measure. Thus,

[2] Conditional probability $P(A \mid B)$ is defined by:

$$P(A \mid B) = \frac{P(A, B)}{P(B)}$$

The chain rule for conditional probability is given by:

$$P(A, B, C, D) = P(A|B, C, D)\ P(B|C, D)\ P(C|D)\ P(D)$$

the probability is given by a Gaussian centred on the prediction $\mathbf{H}\hat{\mathbf{x}}_k$ with innovation covariance $\mathbf{B}$:

$$\mathcal{N}(\mathbf{z}_j - \mathbf{H}\hat{\mathbf{x}}_k, \mathbf{B}) \tag{6.12}$$

In the above equation, $(\mathbf{z}_j - \mathbf{H}\hat{\mathbf{x}}_k)$ is the innovation mentioned earlier. And $\mathbf{H}$ stands for the estimator's implementation of the generic function h_t of Equation 6.4, that maps the state space to the measurement space. $\mathbf{B}$ is the estimator's noise model that incorporates the uncertainty of prediction as well as measurement noise (the noise components $\mathbf{w}(t)$ and $\mathbf{v}(t)$ of Equations 6.1 and 6.4 respectively).

Typically, *measurement gating* is used to eliminate unlikely measurement to track associations [95]. This is in fact an important component of the MHT algorithm as it helps to reduce the total number of hypotheses generated in each time step – see Figure 6.1. Measurement gating defines a region in the measurement space, known as the *validation region*, which is defined as follows [100]:

$$(\mathbf{z}_j - \mathbf{H}\hat{\mathbf{x}}_k)^T \mathbf{B}^{-1} (\mathbf{z}_j - \mathbf{H}\hat{\mathbf{x}}_k) \leq G \tag{6.13}$$

In the equation above, G is the gate threshold. Figure 6.2 shows an example; only measurements that fall within the target's validation region G are associated with it during hypothesis generation [3].

Next we determine the probability of a measurement $\mathbf{z}_j$ being associated with a new track or as false alarm.

It is assumed that the probability of the occurrence of either event is spatially uniform across the sensor's FOV. In other words, a false alarm can occur at any point in the sensor's FOV with the same probability; the same happens for the occurrence of a new target. If the area of the FOV is V, then the probability is equal to $\frac{1}{V}$.

Combining the detection probability of Equation 6.12 with the uniform probability for new and false alarms, we get the complete probability for Factor 1 of Equation 6.10:

[3] Some MHT implementations adopt a coarse gating instead of or in conjunction with the measurement gating described here. This is done for efficiency reasons: for example, a rectangular gating is used by Vo et al. [101].

$$P\Big(Z(t) \mid \psi_i(t), \Omega_{g(i)}^{t-1}, Z^{t-1}\Big) = \prod_{j=1}^{M_t} \underbrace{\mathcal{N}(\mathbf{z}_j - \mathbf{H}\hat{\mathbf{x}}_k, \mathbf{B})^{\delta_j}}_{\text{term 1}} \underbrace{V^{\delta_j - 1}}_{\text{term 2}}$$

$$= V^{-\left(N_{fa}+N_{new}\right)} \prod_{j=1}^{M_t} \mathcal{N}(\mathbf{z}_j - \mathbf{H}\hat{\mathbf{x}}_k, \mathbf{B})^{\delta_j} \tag{6.14}$$

where δ_j is an indicator variable taking the value of 1 if a measurement $\mathbf{z}_j$ has been associated with some target k and 0 otherwise (new track or false alarm). In Equation 6.14 above, either term 1 only or term 2 only can be active for a given measurement $\mathbf{z}_j$. Term 2 does not depend on the measurement's position due to the assumption of spatial uniform probability and thus can be taken out of the product term. There will be $N_{fa} + N_{new}$ occurrences of term 2, where N_{fa} is the number of measurements associated with false alarms and N_{new} is the number of measurements associated with new tracks. Note that the product in Equation 6.14 is based on the assumption that the measurements $\{\mathbf{z}_j\}$ in a scan are statistically independent. Note also that in Equation 6.14 above, we do not show iteration over subscript k in order to simplify the notation: we assume a single measurement $\mathbf{z}_j$ may get associated with a corresponding target k, with predicted state $\hat{\mathbf{x}}_k$, and that no two different measurements match with the same target k.

Moving on to Factor 2 of Equation 6.10, this factor gives the likelihood of a new data association hypothesis at time t, i.e. the probability of assignment $\psi_i(t)$, given the corresponding parent hypothesis. Factor 2 can be sub-divided into three new factors as shown below:

$$P\Big(\psi_i(t) \mid \Omega_{g(i)}^{t-1}, Z^{t-1}\Big) = \underbrace{P\Big(N_{det}, N_{fa}, N_{new} \mid \Omega_{g(i)}^{t-1}, Z^{t-1}\Big)}_{\text{factor 2a}}$$

$$\cdot \underbrace{P(\text{configuration} \mid N_{det}, N_{fa}, N_{new})}_{\text{factor 2b}}$$

$$\cdot \underbrace{P(\text{assignment} \mid \text{configuration})}_{\text{factor 2c}} \tag{6.15}$$

Note that the MHT algorithm makes use of explicit explanations to interpret the observed data, and this is what is happening with the breakdown of terms in Equation 6.15.

Factor 2a of Equation 6.15 gives the probability of obtaining the total counts of targets with a certain label (N_{det}, N_{fa}, N_{new}), given the parent hypothesis. N_{det} is the total number of detected targets, N_{fa} is the number of false alarms, while N_{new} is the number of new targets. Factor 2b gives the probability of a specific configuration of

measurements, given the counts. In other words, the probability of a specific partitioning of the measurements into the three classes according to the given counts. And factor 2c specifies the probability of a specific assignment of tracks to the partitioned measurements [97]. Figure 6.5 illustrates with an example the meaning of these three factors.

Let the current set of measurements detected at time t be:	$\{\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4, \mathbf{z}_5\}$
Consider a particular data association $\psi_i(t)$ with counts:	$N_{det} = 3$ $N_{fa} = 2$ $N_{new} = 0$
Factor 2a: $P\left(N_{det}, N_{fa}, N_{new} \mid \Omega_{g(i)}^{t-1}, Z^{t-1}\right)$:	the probability of having 3 detections and 2 false alarms out of 5 observations: $= P($3 detections out of 5, 2 false alarms out of 5$)$
Factor 2b: $P(\text{configuration} \mid N_{det}, N_{fa}, N_{new})$:	the probability of a specific partitioning of measurements: $= P(\text{detections} = \{\mathbf{z}_1, \mathbf{z}_3, \mathbf{z}_4\},$ false alarms $= \{\mathbf{z}_2, \mathbf{z}_5\})$
Factor 2c: $P(\text{assignment} \mid \text{configuration})$:	the probability of a specific assignment: $= P(\mathbf{z}_1 \rightarrow \mathbf{x}_{15},$ $\mathbf{z}_3 \rightarrow \mathbf{x}_9,$ $\mathbf{z}_4 \rightarrow \mathbf{x}_{23},$ $\mathbf{z}_2 \rightarrow$ fa, $\mathbf{z}_5 \rightarrow$ fa$)$

FIGURE 6.5: Data association likelihood example

Assuming independence between associations, Factor 2a of Equation 6.15 can be expressed as a product of terms:

$$\begin{aligned} &P\left(N_{det}, N_{fa}, N_{new} \mid \Omega_{g(i)}^{t-1}, Z^{t-1}\right) \\ &\quad = P\left(N_{det} \mid \Omega_{g(i)}^{t-1}, Z^{t-1}\right)\ P\left(N_{fa} \mid \Omega_{g(i)}^{t-1}, Z^{t-1}\right)\ P\left(N_{new} \mid \Omega_{g(i)}^{t-1}, Z^{t-1}\right) \end{aligned} \tag{6.16}$$

The probability of detecting N_{det} out of the N targets of the parent hypothesis, is assumed to follow a binomial distribution: $\mathcal{B}(N, p_{det})$, where parameter N is the total number of tracks (number of *trials* for the binomial), and parameter $p_{det} \in [0, 1]$ is the detection success probability for each trial (each track). Thus:

$$P\left(N_{det} \mid \Omega_{g(i)}^{t-1}, Z^{t-1}\right) = P_{\mathcal{B}}(N_{det}; N, p_{det}) = \binom{N}{N_{det}} p_{det}^{N_{det}} \left(1 - p_{det}\right)^{N - N_{det}} \tag{6.17}$$

In the above equation, $\binom{N}{N_{det}}$ is the binomial coefficient [4].

It is assumed that the probability of false alarms and the probability of new tracks follow Poisson distributions $\mathcal{P}(\lambda_{fa}V)$ and $\mathcal{P}(\lambda_{new}V)$ respectively, with λ_{fa} and λ_{new} being the *event rates* of the Poisson distributions for a point in the observation space. The event rates need to be multiplied by the observation volume V (the sensor's FOV), because the larger the volume, the greater is the chance of false alarm occurrences and the greater is the chance of new targets entering the FOV. Thus:

$$P\Big(N_{fa} \,|\, \Omega_{g(i)}^{t-1}, Z^{t-1}\Big) = P(N_{fa}; \lambda_{fa}V) = \frac{(\lambda_{fa}V)^{N_{fa}}}{N_{fa}!} e^{-\lambda_{fa}V} \tag{6.18}$$

$$P\Big(N_{new} \,|\, \Omega_{g(i)}^{t-1}, Z^{t-1}\Big) = P(N_{new}; \lambda_{new}V) = \frac{(\lambda_{new}V)^{N_{new}}}{N_{new}!} e^{-\lambda_{new}V} \tag{6.19}$$

Substituting Equations 6.17, 6.18 and 6.19 into Equation 6.16 gives us:

$$P\Big(N_{det}, N_{fa}, N_{new} \,|\, \Omega_{g(i)}^{t-1}, Z^{t-1}\Big) = \binom{N}{N_{det}} p_{det}^{N_{det}} \left(1 - p_{det}\right)^{N - N_{det}} \frac{(\lambda_{fa}V)^{N_{fa}}}{N_{fa}!} e^{-\lambda_{fa}V} \frac{(\lambda_{new}V)^{N_{new}}}{N_{new}!} e^{-\lambda_{new}V} \tag{6.20}$$

Factor 2b of Equation 6.15 gives us the probability of a specific configuration of the M_t measurements at time t, which is subject to the following self-evident constraint:

$$M_t = N_{det} + N_{fa} + N_{new} \tag{6.21}$$

The probabilty of a configuration depends on the total number of possible ways in which we can partition the measurements into detected targets, false alarms and new targets. Using combinatorials, the total number of choices can be expressed as:

$$N_{\text{configurations}} = \binom{M_t}{N_{det}} \binom{M_t - N_{det}}{N_{fa}} \binom{M_t - N_{det} - N_{fa}}{N_{new}} \tag{6.22}$$

In Equation 6.22 above, the last term is 1 since it follows from Equation 6.21 that $M_t - N_{det} - N_{fa} = N_{new}$; i.e., the remaining measurements are new tracks by default.

[4] The binomial coefficient:

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

gives the number of different ways we can choose k items out of n, where the order of selection of the k items does not matter.

Equation 6.22 can be described as: out of a total of M_t measurements, we can choose N_{det} measurements to associate with targets; out of the remaining measurements, we choose N_{fa} of them to be false targets, and the remaining N_{new} measurements are new targets [96]. Factor 2b of Equation 6.15 is therefore given by:

$$P(\text{configuration}|N_{det}, N_{fa}, N_{new}) = \frac{1}{N_{\text{configurations}}} = \frac{1}{\binom{M_t}{N_{det}}\binom{M_t - N_{det}}{N_{fa}}} \tag{6.23}$$

Finally, for a specific configuration, the total number of ways of permuting the assignments of N_{det} measurements to the N tracks is given by:

$$N_{\text{assignments}} = \frac{N!}{(N - N_{det})!} \tag{6.24}$$

and the probability of Factor 2c (Equation 6.15) is given by:

$$P(\text{assignment}|\text{configuration}) = \frac{1}{N_{\text{assignments}}} = \frac{(N - N_{det})!}{N!} \tag{6.25}$$

Substituting Equations 6.20, 6.23 and 6.25 into Equation 6.15, and re-arranging the terms, we get:

$$\frac{(N - N_{det})!}{N!} \frac{1}{\binom{M_t}{N_{det}}\binom{M_t - N_{det}}{N_{fa}}} \binom{N}{N_{det}} p_{det}^{N_{det}} (1 - p_{det})^{N - N_{det}} \cdot \frac{(\lambda_{fa} V)^{N_{fa}}}{N_{fa}!} e^{-\lambda_{fa} V} \frac{(\lambda_{new} V)^{N_{new}}}{N_{new}!} e^{-\lambda_{new} V}$$

Expanding the binomial coefficients and simplifying:

$$\frac{\cancel{(N - N_{det})!}}{\cancel{N!}} \frac{\cancel{N_{det}!}\,\cancel{(M_t - N_{det})!}}{M_t!} \frac{N_{fa}!\,(M_t - N_{det} - N_{fa})!}{\cancel{(M_t - N_{det})!}} \frac{\cancel{N!}}{\cancel{N_{det}!}\,\cancel{(N - N_{det})!}} \cdot p_{det}^{N_{det}} (1 - p_{det})^{N - N_{det}} \frac{\lambda_{fa}^{N_{fa}} V^{N_{fa}}}{N_{fa}!} e^{-\lambda_{fa} V} \frac{\lambda_{new}^{N_{new}} V^{N_{new}}}{N_{new}!} e^{-\lambda_{new} V}$$

We can observe that many of the terms cancel out. According to Equation 6.21, we can replace $(M_t - N_{det} - N_{fa})$ with N_{new}. Substituting and simplifying further:

$$\frac{\cancel{N_{fa}!}\,\cancel{N_{new}!}}{M_t!} p_{det}^{N_{det}} (1 - p_{det})^{N - N_{det}} \frac{\lambda_{fa}^{N_{fa}} V^{N_{fa}}}{\cancel{N_{fa}!}} e^{-\lambda_{fa} V} \frac{\lambda_{new}^{N_{new}} V^{N_{new}}}{\cancel{N_{new}!}} e^{-\lambda_{new} V}$$

Re-arranging and moving the constants to the front, we end up with:

$$P\Big(\psi_i(t) \mid \Omega_{g(i)}^{t-1}, Z^{t-1}\Big) = \underbrace{\frac{e^{-\lambda_{fa}V} e^{-\lambda_{new}V}}{M_t!}}_{\text{constants}} \; p_{det}^{N_{det}} \left(1 - p_{det}\right)^{N - N_{det}} \lambda_{fa}^{N_{fa}} V^{N_{fa}} \lambda_{new}^{N_{new}} V^{N_{new}} \tag{6.26}$$

Note that the constants in Equation 6.26 are independent of the chosen assignment $\psi_i(t)$. Finally, substituting Equations 6.14 and 6.26 in to the original Equation 6.10, we get:

$$\begin{aligned} P\big(\Omega_i^t \mid Z^t\big) &= \frac{1}{c}\, P\Big(Z(t) \mid \psi_i(t), \Omega_{g(i)}^{t-1}, Z^{t-1}\Big)\; P\Big(\psi_i(t) \mid \Omega_{g(i)}^{t-1}, Z^{t-1}\Big)\; P\Big(\Omega_{g(i)}^{t-1} \mid Z^{t-1}\Big) \\ &= \frac{1}{c}\, \frac{1}{\cancel{V^{N_{fa}}}\,\cancel{V^{N_{new}}}} \prod_{j=1}^{M_t} \mathcal{N}(\mathbf{z}_j - \mathbf{H}\hat{\mathbf{x}}_k, \mathbf{B})^{\delta_j} \\ &\quad \cdot\, \frac{e^{-\lambda_{fa}V} e^{-\lambda_{new}V}}{M_t!}\, p_{det}^{N_{det}} \left(1 - p_{det}\right)^{N - N_{det}} \lambda_{fa}^{N_{fa}} \cancel{V^{N_{fa}}} \lambda_{new}^{N_{new}} \cancel{V^{N_{new}}} \\ &\quad \cdot\, P\Big(\Omega_{g(i)}^{t-1} \mid Z^{t-1}\Big) \end{aligned}$$

which can be simplified further by grouping the constants together [5] into c:

$$P\big(\Omega_i^t \mid Z^t\big) = \frac{1}{c} \prod_{j=1}^{M_t} \mathcal{N}(\mathbf{z}_j - \mathbf{H}\hat{\mathbf{x}}_k, \mathbf{B})^{\delta_j}\; p_{det}^{N_{det}} \left(1 - p_{det}\right)^{N - N_{det}}\; \lambda_{fa}^{N_{fa}} \lambda_{new}^{N_{new}}\; P\Big(\Omega_{g(i)}^{t-1} \mid Z^{t-1}\Big)$$

By dropping the indicator variable δ_j and iterating only over the N_{det} detected measurements, and after re-arranging the terms a bit, we get the following hypothesis likelihood equation:

$$\begin{aligned} P\big(\Omega_i^t \mid Z^t\big) = \; & \frac{1}{c}\, p_{det}^{N_{det}} \left(1 - p_{det}\right)^{N - N_{det}} \lambda_{fa}^{N_{fa}} \lambda_{new}^{N_{new}} \\ & \cdot \prod_{j=1}^{N_{det}} \mathcal{N}(\mathbf{z}_j - \mathbf{H}\hat{\mathbf{x}}_k, \mathbf{B})\; P\Big(\Omega_{g(i)}^{t-1} \mid Z^{t-1}\Big) \end{aligned} \tag{6.27}$$

The above equation is the same as Equation 5.4 (main PhD thesis report, §5.3.1, p. 104).

[5] To simplify, we do not change the name of the constant term c in the equation even though it is now different. For our purposes, it is still a constant term.

6.4 Extending the MHT algorithm – examples from the Literature

In this section, we briefly describe a couple of extensions that have been applied to the original MHT algorithm of Reid [96]. And in the process, we show how amenable to change the formulation of the MHT is. Table 6.1 below summarises these works.

Paper	Concept	Affected part	Extensions to hypothesis likelihood equation (changes in red)
Cox and Hingorani [80]	Track deletion	factor 1	$p_{del}{}^{n}(1-p_{del})^{N-n}$
Arras et al. [97]	Occlusion	factor 1	$p_{occ}{}^{n}(1-p_{occ})^{N-n}$
Luber et al. [102]	Spatially varying false alarm and new target rates	factor 2	$\mathcal{N}(\cdot)^{\delta_1}\,\lambda_{fa}(z_j)^{\delta_2}\,\lambda_{new}(z_j)^{\delta_3}$, where δ_i are indicator variables
Wyffels and Campbell [103]	"Negative observations" to better model undetected and occluded objects	factor 2	$\mathcal{N}(\cdot)\,L_{occ}$
Coraluppi and Carthel [98]	Reward factors for nearly-confirmed and confirmed tracks	factor 2	$\mathcal{N}(\cdot)^{\delta_1}\;\xi_1^{\delta_2}\;\xi_2^{\delta_3}$, where δ_i are indicator variables
Zúñiga et al. [104]	Temporal reliability measure T_j based on object's lifetime	factor 2	$\mathcal{N}(\cdot)\,T_j$
Ying et al. [105]	Integrate appearance similarity score based on HSV + LBP	factor 2	$(1-\alpha)\,\mathcal{N}(\cdot) + \alpha\,P_{appearance}\,(Z^t)$
Manafifard et al. [106]	Appearance models, plus label consistency term	factor 2	$\mathcal{N}(\cdot)\,\lambda^{L}$
Antunes et al. [107]	Appearance information, plus surveillance zone in a multi-camera system	factor 2	$\mathcal{N}(\cdot)\,P_{appearance}\left(hist^{Z}, hist^{T}\right)$ $P_{transition}\left(zone^{Z}, zone^{T}\right)$

TABLE 6.1: Some extensions to the MHT algorithm

In the standard MHT formulation, the rates of false alarm events and new target events are assumed to follow Poisson distributions temporally, and to spatially occur uniformly over the sensor's FOV. While this is a valid assumption for traditional radar-based applications, in visual tracking the events are normally non-homogeneous in both time and space. The way object and people tracks behave can depend on environmental and scene conditions, which can be learnt (e.g. in an indoor scene, new tracks are more

likely to appear and die near doors. And areas of clutter and complex backgrounds tend to give rise to more false alarms) [102].

Luber et al. [102] learn spatial maps (which they term *spatial affordance maps*) of long-term human activity events in a scene. Then, a Poisson distribution is learnt from the spatial maps, and is used to model the rate of occurrence of events dependant on the position x in volume V. Thus the probability of false alarms, which is specified as a uniform distribution over V in the regular MHT, is replaced by:

$$p_{fa}(x) = \frac{\lambda_{fa}(x,t)}{\lambda_{fa}(t)} = \frac{\lambda_{fa}(x,t)}{\int_V \lambda_{fa}(x,t)\,dx} \approx \frac{\lambda_{fa}(x)}{\int_V \lambda_{fa}(x)\,dx} \tag{6.28}$$

for some position x in the volume V. The last term in Equation 6.28 is based on the simplification that the Poisson process is only non-homogeneous in space, and assumed to be fixed in time, i.e., the false alarm rate varies according to scene position x, but not in time. $\lambda_{fa}(x)$ is the learnt spatial map of false alarm event rates for positions $x \in V$. The denominator is a normalisation term with respect to the sensor volume V. The same formulation is adopted for p_{new}, the probability of new tracks. The modified hypothesis likelihood equation now becomes:

$$\begin{aligned} P\left(\Omega_i^t \mid Z^t\right) = \frac{1}{c}\, & p_{det}^{N_{det}}\,(1-p_{det})^{N-N_{det}} \\ & \cdot \prod_{j=1}^{M_t} \left\{ \mathcal{N}(\mathbf{z}_j - \mathbf{H}\hat{\mathbf{x}}_k, \mathbf{B})^{\delta_j}\, \lambda_{fa}(z_j)^{\phi_j}\, \lambda_{new}(z_j)^{\nu_j} \right\} P\left(\Omega_{g(i)}^{t-1} \mid Z^{t-1}\right) \end{aligned} \tag{6.29}$$

A limitation of the p_{del} term of Cox and Hingorani [80] and the p_{occ} term of Arras et al. [97] is that both methods do not consider the geometry of the scene in track deletion and track occlusion labelling. The scene can provide quite informative indicators of track deletion and track occlusion [103].

Wyffels and Campbell [103] introduce the concept of **"negative observations"** to the MHT. These are artificially generated by the occlusion model whenever a missed detection due to occlusion is inferred, and allow for maintaining of a sensible belief of occluded object states in the tracking hypotheses. Being unobserved, these negative observations have more uncertainty than "positive" observations. The occlusion likelihood function L_{occ} used by Wyffels and Campbell [103] is designed specifically for lidar data and takes into account the number of targets, their geometry and relative occlusion fronts

of the range data. L_{occ} is then added into Equation 6.14, giving rise to:

$$P\Big(Z(t) \mid \psi_i(t), \Omega_{g(i)}^{t-1}, Z^{t-1}\Big) = V^{-\left(N_{fa}+N_{new}\right)} \prod_{j=1}^{M_t} \mathcal{N}(\mathbf{z}_j - \mathbf{H}\hat{\mathbf{x}}_k, \mathbf{B})^{\delta_j} \prod_{k=1}^{N} L_{occ}\left(x_k\right)^{\omega_k} \tag{6.30}$$

where ω_k is an indicator variable having value 1 when target $\mathbf{x}_k$ is considered as occluded and 0 otherwise.

Coraluppi and Carthel [98] add reward factors $\xi_1 > \xi_2 > 1$ to confirmed and nearly-confirmed tracks in order to solve the problem whereby unconfirmed tracks "steal" observations from confirmed or nearly-confirmed tracks. These factors are added as multipliers to Factor 2 of Equation 6.27.

Zúñiga et al. [104] introduce a temporal reliability measure T_j to Equation 6.27, taking into account the lifetime of the target. The result is a weighting of the target's dynamical motion model with the proposed reliability measure:

$$\mathcal{N}(\mathbf{z}_j - \mathbf{H}\hat{\mathbf{x}}_k, \mathbf{B})\, T_j \tag{6.31}$$

where T_j is computed based on the number of frames since an object j has been seen for the first time. They also incorporate 2D and 3D visual attribute features into the hypothesis probability; the former based on the 2D blob geometry (width and height), and the latter obtained by fitting a 3D parallelepiped model (width, height, volume, groundplane rotation) to the image data.

Perhaps a critique that can be made of the reward factor proposed by Coraluppi and Carthel [98] and the temporal reliability factor proposed by Zúñiga et al. [104] is that they are essentially "fudge" factors. Such factors tend to be ad hoc adjustable parameters with weak theoretical justification. Typically, such parameters end up being set to conservative values, determined empirically [93]. Since one of the main strengths of the MHT algorithm is its strong statistical foundation, using such factors is not ideal. Incorporating statistically sound uncertainty and error quantities within the Bayesian formulation of MHT is preferred [93].

Traditionally, the MHT technique arose in the radar literature, where objects are observed at a distance and are considered to be point sources. In imaging type sensors, targets form an 'extended' source, and additional (non-kinematic) feature-based information about a target is available, e.g., shape, intensity, colour. The use of such features is sometimes referred to as *feature-aided tracking* (if the features are used to augment the target state), and *classification-aided tracking* (if a database exists that links features

to target types) [108]. Such features are used to improve gating and/or data association during tracking.

Ying et al. [105] extend the MHT formulation to integrate appearance information (HSV histograms and local binary patterns (LBP) histograms). They modify Factor 2 of Equation 6.27 to augment the kinematic scores of the classical MHT with the addition of HSV+LBP histogram similarity scores. These are combined as a weighted sum of both components:

$$(1-\alpha)\,\mathcal{N}(\cdot) + \alpha\, P_{appearance}\left(Z^t \mid \tau_g^{t-1}\right) \tag{6.32}$$

where τ_g^{t-1} is the template of the target, and $P_{appearance}$ uses the Bhattacharrya metric transformed to a probability value via the logistic function. They also consider a repulsion-inertial model for handling objects in proximity to estimate their states and better handle identity switches.

A critique that can be made of methods that do a weighted combination of the track likelihood score is that this does not follow a mathematically rigorous model-based evidence accrual. The combination is not done objectively, and depends on subjective weights. The likelihood scores (e.g. kinematic score and appearance score), may be quite dissimilar in nature [93]. A more statistically sound way of combining the dissimilar data is needed, that better reflects the accumulation of evidence (information) and the proper aggregation of uncertainty [93].

Manafifard et al. [106] also include appearance-based information within the MHT, which is used for tracking football players. The use of appearance information proved to be especially helpful during merge events. For ambiguous measurements (during merge/partial occlusion events), they uses a particle swarm optimisation search technique and pre-defined colour models to get a "corrected" measurement. This corrected measurement $\mathbf{z}_j^c$ replaces the original measurement $\mathbf{z}_j$ when computing Factor 2 of Equation 6.27. They also add a label consistency term λ^L to it:

$$\mathcal{N}(\cdot)\,\lambda^L \tag{6.33}$$

Manafifard et al. [106] chose the appearance models for the targets manually (team player colours).

Kim et al. [109] incorporate long-term appearance modelling into the MHT, using features obtained from deep CNNs. The computational cost of the training of the appearance models has little dependency on the number of hypothesis branches, thus making the modified MHT as efficient as the standard version. The appearance model

is trained and updated for each target over time. Then they use least squares regression as classifiers and for each hypothesis they train with positive samples taken from the hypothesised track and the rest as negative samples. Their reasoning is that the correct hypothesis should produce a more consistent classifier than a wrong hypothesis. The final track score is thus a weighted combination of the motion information and the best score obtained by the most consistent classifier when trained on the appearance information.

There is also a large body of work looking at classical MHT and multi-stage MHT variants applied to multi-sensor networks, especially those with different types of sensors, such as having a mix of low-rate and high-rate sensors [99, 107, 110, 111].

Antunes et al. [107] perform people tracking with the MHT algorithm across camera networks. They extend the target state information by adding the target ID, surveillance zone ID, histograms of the lower and upper halves of the targets, and time of last detection. Factor 2 of Equation 6.27 is extended to include the additional factors:

$$\mathcal{N}(\cdot) \cdot P\left(\text{hist}^Z, \text{hist}^T\right) \cdot P\left(\text{zone}^Z, \text{zone}^T\right) \tag{6.34}$$

where $P\left(\text{hist}^Z, \text{hist}^T\right)$ gives the histogram similarity (using Bhattacharya coefficient) between that of the target and the measurement, and $P\left(\text{zone}^Z, \text{zone}^T\right)$ gives the probability of a target transitioning from one surveillance zone (covered by one camera) to another zone (covered by a second camera). This latter probability is set to 1 when the two zones are the same, and set to some pre-defined constant k when the surveillance zones are connected together (e.g. via a door or an intermediate zone). Zone connections are defined *a priori* and expressed as a graph structure. Note that in the above equation, the assumption is that the kinematic space data (term 1), appearance data (term 2), and zone positions are statistically independent, rather than considering them as a joint probability density function (pdf).

Several works have also looked into incorporating **target type data** (target **classification data**) into the MHT algorithm. Target type can be incorporated into the gating mechanism, by pairing only targets and measurements of the same class type. Or directly incorporating the target classification scores in factor two of the hypothesis likelihood equation. Notable examples of such works include Wigren [112], and Pannetier and Dezert [113].

Target behaviour classification is also incorporated into MHT in a similar manner. While target type data can be determined from a single scan, target behaviour classification is normally performed over a time window [114]. Both types of data can be incorporated with the kinematic data being fed to the MHT or treated as the 'data' itself

[114]. For target type data, term $P\left(Z(t) \mid \psi_i(t), \Omega_{g(i)}^{t-1}, Z^{t-1}\right)$ in Equation 6.10 can be expressed as follows for each individual measurement:

$$P(\mathbf{z}_j(t) \mid \psi_i(t), \Omega_{g(i)}^{t-1}, Z^{t-1}) = P\left(\mathbf{z}_{ks,j}(t) \mid \psi_i(t), \Omega_{g(i)}^{t-1}, Z^{t-1}\right) \cdot P\left(\mathbf{z}_{tt,j}(t) \mid \psi_i(t), \Omega_{g(i)}^{t-1}, Z^{t-1}\right) \tag{6.35}$$

where $\mathbf{z}_{ks,j}$ represents the kinematic state of the measurement and $\mathbf{z}_{tt,j}$ is the target type data of the measurement. The product of the two terms is based on the assumption that the kinematic space data and the target type feature space data are statistically independent. Examples of such works include Pannetier and Dezert [113], Blasch [115], Lancaster and Blackman [116], and Bendich et al. [114].

6.5 MHT with hand motion constraints – additional results

In this section, we provide some additional qualitative results for the experiments conducted in §5.6 of the main PhD thesis report.

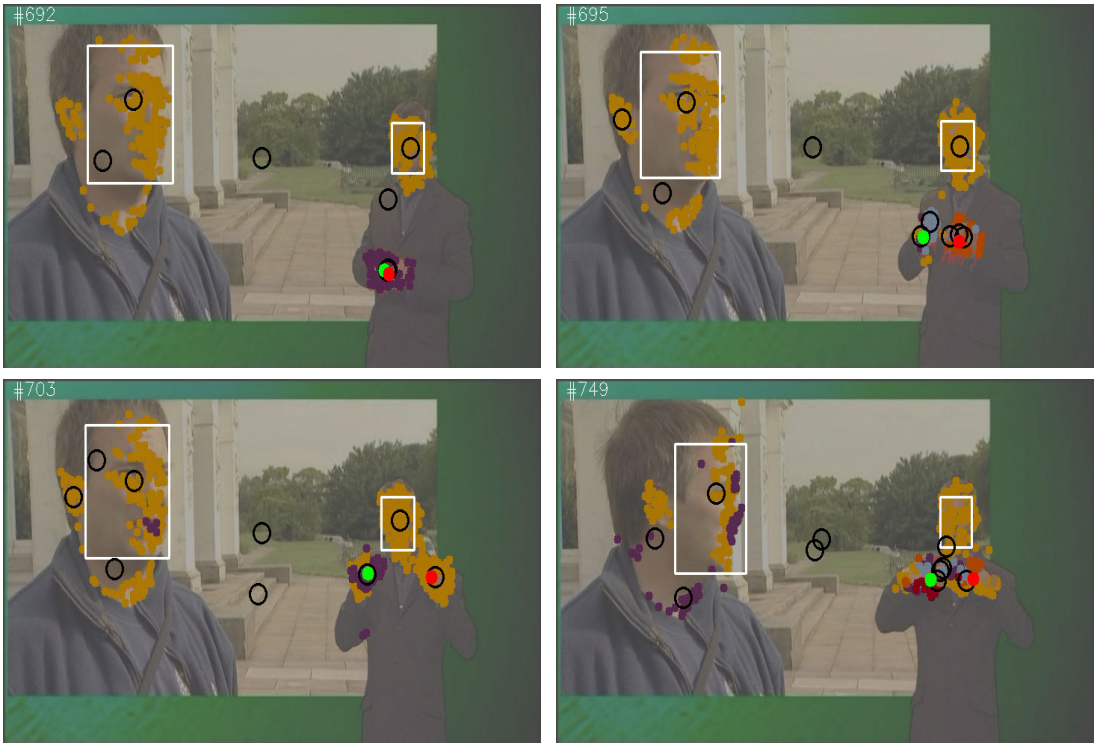
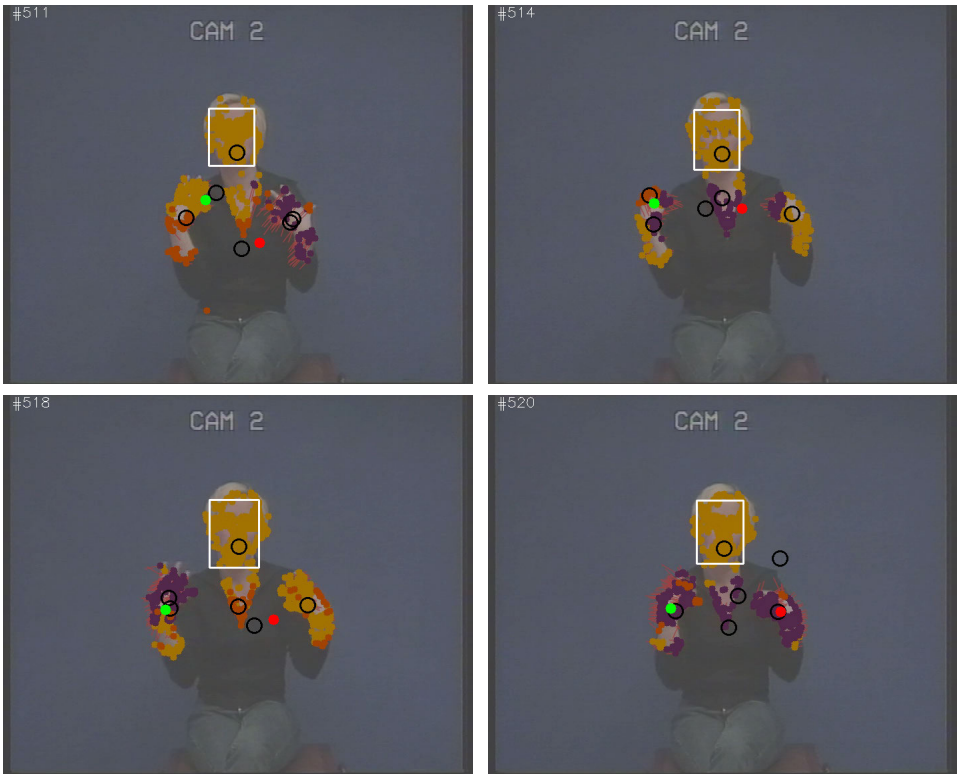
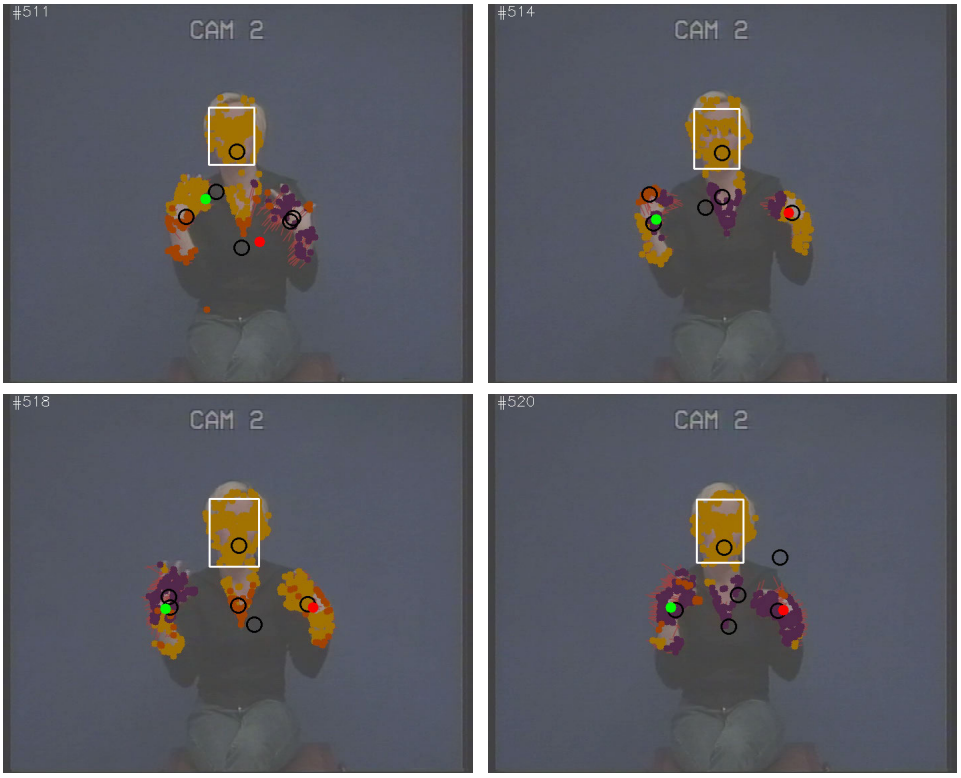


FIGURE 6.6: Example of SL hand motion constraints applied to the BBC Pose dataset, video sequence #1



(A) Unsuccessful hand tracking of a symmetric sign with the "baseline MHT". The sign is Dutch sign language (Nederlandse Gebarentaal, NGT) SHEEP.



(B) The same symmetric sign successfully tracked with our "MHT + SL constraints" method

FIGURE 6.7: Example of hand tracking with SL hand motion constraints

References

[1] Viola, P. and Jones, M. J. Robust Real-time Object Detection. *Int. J. of Computer Vision*, 2001.

[2] Viola, P. and Jones, M. Robust real-time face detection. *Int. J. of Computer Vision*, 57(2):137–154, 2004.

[3] Lienhart, R., Kuranov, A., and Pisarevsky, V. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *DAGM 25th Pattern Recognition Symposium*, pp. 297–304, 2003.

[4] Bradski, G. R. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[5] Schapire, R. E. and Singer, Y. Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, 37(3):297–336, 1999. ISSN 1573-0565.

[6] Jones, M. and Viola, P. Fast multi-view face detection. Technical report, Mitsubishi Electric Research Lab, 2003.

[7] Danisman, T. and Bilasco, I. M. In-plane Face Orientation Estimation in Still Images. *Multimedia Tools Appl.*, 75(13):7799–7829, 2016. ISSN 1380-7501.

[8] Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, October 2016. ISSN 1070-9908.

[9] Chollet, F. et al. Keras. 2015. URL https://keras.io.

[10] Abadi, M., Agarwal, A., Barham, P., et al. TensorFlow: Large-scale machine learning on heterogeneous systems. 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[11] Dalitz, C., Pohle-Fröhlich, R., Schmitt, F., and Jeltsch, M. The gradient product transform for symmetry detection and blood vessel extraction. In *VISAPP 2015 - Proc. 10th Int. Conf. on Computer Vision Theory and Applications*, volume 2, pp. 177–184, 2015.

[12] Perez-Sala, X., Escalera, S., Angulo, C., and Gonzàlez, J. A survey on model based approaches for 2d and 3d visual human pose recovery. *Sensors*, 14(3): 4189–4210, 2014. doi: 10.3390/s140304189.

[13] Greenspan, H., Goldberger, J., and Eshet, I. Mixture model for face-color modeling and segmentation. *Pattern Recognition Letters*, 22(14):1525–1536, 2001. ISSN 0167-8655. doi: 10.1016/S0167-8655(01)00086-1.

[14] Vezhnevets, V., Sazonov, V., and Andreeva, A. A survey on pixel-based skin color detection techniques. In *Proc. GRAPHICON*, pp. 85–92, 2003.

[15] Wimmer, M., Radig, B., and Beetz, M. A person and context specific approach for skin color classification. In *Proc. ICPR*, volume 2, pp. 39–42, 2006.

[16] Kakumanu, P., Makrogiannis, S., and Bourbakis, N. A survey of skin-color modeling and detection methods. *Pattern Recogn.*, 40(3):1106–1122, 2007.

[17] Kawulok, M. Dynamic skin detection in color images for sign language recognition. In *Proc. 3rd Int. Conf. on Image and Signal Processing*, pp. 112–119. Springer-Verlag, 2008. ISBN 978-3-540-69904-0.

[18] Han, J., Awad, G., and Sutherland, A. Automatic skin segmentation and tracking in sign language recognition. *Computer Vision, IET*, 3(1):24–35, 2009.

[19] Liensberger, C., Stöttinger, J., and Kampel, M. Color-based and context-aware skin detection for online video annotation. *2009 IEEE Int. Workshop on Multimedia Signal Processing*, 2009.

[20] Shaik, K. B., Ganesan, P., Kalist, V., Sathish, B., and Jenitha, J. M. M. Comparative study of skin color detection and segmentation in HSV and YCbCr color space. *Procedia Computer Science*, 57:41–48, 2015. ISSN 1877-0509.

[21] Gomez, G. and Morales, E. F. Automatic Feature Construction and a Simple Rule Induction Algorithm for Skin Detection. In *Proc. ICML Workshop on Machine Learning in Computer Vision*, pp. 31–38, 2002.

[22] Lee, J. Y. and Yoo, S. I. An elliptical boundary model for skin color detection. In *In Proc. Int. Conf. on Imaging Science, System and Technology*, 2002.

[23] Shoyaib, M., Abdullah-Al-Wadud, M., and Chae, O. A skin detection approach based on the dempster–shafer theory of evidence. *Int. Journal of Approximate Reasoning*, 53(4):636–659, 2012. doi: 10.1016/j.ijar.2012.01.003.

[24] Albiol, A., Torres, L., and Delp, E. J. Optimum color spaces for skin detection. In *Proc. ICIP*, volume 1, pp. 122–124, 2001. doi: 10.1109/ICIP.2001.958968.

[25] Elgammal, A., Muang, C., and Hu, D. Skin detection - a short tutorial. *Encyclopedia of Biometrics*, 2009.

[26] Wimmer, M. and Radig, B. Adaptive skin color classificator. In *Proc. 1st ICGST Int. Conf. on Graphics, Vision and Image Processing (GVIP)*, p. 324, 2005.

[27] Cheddad, A., Condell, J., Curran, K., and Kevitt, P. M. A skin tone detection algorithm for an adaptive approach to steganography. *Signal Processing*, 89(12): 2465–2478, 2009. ISSN 0165-1684. doi: 10.1016/j.sigpro.2009.04.022.

[28] Deng, Y. and Manjunath, b. s. Unsupervised segmentation of color-texture regions in images and video. *IEEE TPAMI*, 23(8):800–810, 2001.

[29] Wang, B., Chang, X., and Liu, C. A robust method for skin detection and segmentation of human face. In *2nd Int. Conf. on Intelligent Networks and Intelligent Systems*, p. 290, November 2009. doi: 10.1109/ICINIS.2009.80.

[30] Kawulok, M., Kawulok, J., Nalepa, J., and Smolka, B. Self-adaptive algorithm for segmenting skin regions. *EURASIP J. on Advances in Signal Processing*, 2014(1): 170, November 2014. ISSN 1687-6180. doi: 10.1186/1687-6180-2014-170.

[31] Lucas, B. D. and Kanade, T. An iterative image registration technique with an application to stereo vision. In *Proc. 7th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 674–679, 1981.

[32] Tomasi, C. and Kanade, T. Detection and tracking of point features. Technical Report Technical Report CMU-CS-91-132, Carnegie Mellon University, 1991. Technical Report CMU-CS-91-132, April 1991.

[33] Shi, J. and Tomasi, C. Good features to track. In *Proc. CVPR*, pp. 593–600, 1994.

[34] Tuytelaars, T. and Mikolajczyk, K. Local Invariant Feature Detectors: A Survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, 2008.

[35] Li, Y., Wang, S., Tian, Q., and Ding, X. A survey of recent advances in visual feature detection. *Neurocomputing*, 149, Part B:736–751, 2015. ISSN 0925-2312.

[36] Govender, N. Evaluation of feature detection algorithms for structure from motion. Technical report, Mobile Intelligent Autonomous Systems, CSIR, Pretoria, 2009.

[37] Fleet, D. and Weiss, Y. Optical flow estimation. In Paragios, N. et al., editors, *Handbook of Mathematical Models in Computer Vision*, chapter 15, pp. 237–257. Springer, 2006. doi: 10.1007/0-387-28831-7_15.

[38] Thacker, N. A., Clark, A. F., Barron, J. L., Beveridge, J. R., Courtney, P., Crum, W. R., Ramesh, V., and Clark, C. Performance characterization in computer vision: A guide to best practices. *Computer Vision and Image Understanding*, 109(3): 305–334, 2008. ISSN 1077-3142. doi: 10.1016/j.cviu.2007.04.006.

[39] Horn, B. K. and Schunck, B. G. Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203, 1981. ISSN 0004-3702. doi: 10.1016/0004-3702(81)90024-2.

[40] Baker, S. and Matthews, I. Lucas-Kanade 20 Years On: A Unifying Framework. *Int. J. of Computer Vision*, 56(3):221–255, February 2004. ISSN 1573-1405.

[41] Harris, C. and Stephens, M. A combined corner and edge detector. In *Proc. of Fourth Alvey Vision Conference*, pp. 147–151, 1988.

[42] Bouguet, J.-Y. Pyramidal Implementation of the Lucas-Kanade Feature Tracker. 2000.

[43] Sheorey, S., Keshavamurthy, S., Yu, H., Nguyen, H., and Taylor, C. N. Uncertainty estimation for KLT tracking. In *Proc. ACCV*, pp. 475–487. Springer, Cham, 2015. ISBN 978-3-319-16631-5. doi: 10.1007/978-3-319-16631-5_35.

[44] Birchfield, S. KLT: An Implementation of the Kanade-Lucas-Tomasi Feature Tracker. August 2007. URL https://cecas.clemson.edu/~stb/klt/. Version 1.3.4.

[45] Ramakrishnan, N., Srikanthan, T., Lam, S. K., and Tulsulkar, G. R. Adaptive Window Strategy for High-Speed and Robust KLT Feature Tracker. In *Image and Video Technology: 7th Pacific-Rim Symposium (PSIVT)*, pp. 355–367. Springer, 2016. ISBN 978-3-319-29451-3.

[46] Kanazawa, Y. and Kanatani, K. Do we really have to consider covariance matrices for image features? In *Proc. ICCV*, volume 2, pp. 301–306, 2001.

[47] Sun, Z., Ramesh, V., and Tekalp, A. Error characterization of the factorization method. *Computer Vision and Image Understanding*, 82(2):110–137, 2001. ISSN 1077-3142. doi: 10.1006/cviu.2001.0910.

[48] Brooks, M. J., Chojnacki, W., Gawley, D., and van den Hengel, A. What value covariance information in estimating vision parameters? In *Proc. ICCV*, volume 1, pp. 302–308, 2001. doi: 10.1109/ICCV.2001.937533.

[49] Choi, S., Kim, T., and Yu, W. Performance evaluation of RANSAC family. In *Proc. BMVC*, 2009.

[50] Zuliani, M. RANSAC for Dummies. Technical report, University of California Santa Barbara, January 2014.

[51] Chum, O. and Matas, J. Matching with PROSAC - progressive sample consensus. In *Proc. CVPR*, pp. 220–226. IEEE, 2005. doi: 10.1109/CVPR.2005.221.

[52] Raguram, R., Frahm, J.-M., and Pollefeys, M. A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus. In *Proc. ECCV*, pp. 500–513. Springer, 2008. ISBN 978-3-540-88688-4.

[53] Huynh, D. Q. and Heyden, A. Outlier detection in video sequences under affine projection. In *Proc. CVPR*, volume 1, pp. 695–701, 2001.

[54] Wu, H., Chellappa, R., Sankaranarayanan, A. C., and Zhou, S. K. Robust visual tracking using the time-reversibility constraint. In *Proc. ICCV*, 2007.

[55] Hedborg, J., Forssén, P.-E., and Felsberg, M. Fast and accurate structure and motion estimation. In *Advances in Visual Computing: 5th Int. Symposium (ISVC)*, pp. 211–222. Springer, 2009. ISBN 978-3-642-10331-5.

[56] Wong, X. I. and Majji, M. Uncertainty Quantification of Lucas Kanade Feature Track and Application to Visual Odometry. In *Proc. CVPR Workshops*, pp. 950–958, July 2017. doi: 10.1109/CVPRW.2017.131.

[57] Sroba, L. and Ravas, R. Sensitivity of subpixel corner detection. In Katalinic, B., editor, *Annals of DAAAM for 2012 & Proc. 23rd Int. DAAAM Symposium*, volume 23, pp. 743–746, 2012.

[58] Thirde, D., Borg, M., Aguilera, J., Wildenauer, H., Ferryman, J., and Kampel, M. Robust Real-Time Tracking for Visual Surveillance. *EURASIP J. on Advances in Signal Processing*, 2007(1), 2007. ISSN 1687-6180.

[59] Borg, M., Thirde, D., Ferryman, J., Fusier, F., Valentin, V., Bremond, F., and Thonnat, M. Video surveillance for aircraft activity monitoring. In *IEEE Conf. on Advanced Video and Signal Based Surveillance*, pp. 16–21, September 2005. doi: 10.1109/AVSS.2005.1577236.

[60] Klicnar, L. and Beran, V. Robust motion segmentation for on-line application. In *Proc. WSCG*, 20th Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision, pp. 205–212. University of West Bohemia in Pilsen, 2012. ISBN 978-80-86943-79-4.

[61] Sivic, J., Schaffalitzky, F., and Zisserman, A. Object level grouping for video shots. In *Proc. ECCV*, pp. 85–98. Springer, 2004. ISBN 978-3-540-24671-8.

[62] Sivic, J., Schaffalitzky, F., and Zisserman, A. Object level grouping for video shots. *Int. J. of Computer Vision*, 67(2):189–210, April 2006. ISSN 1573-1405.

[63] Tanathong, S. and Banharnsakun, A. Multiple Object Tracking Based on a Hierarchical Clustering of Features Approach. In Nguyen, N. T. et al., editors, *Intelligent Information and Database Systems*, pp. 522–529, Cham, 2014. Springer. ISBN 978-3-319-05476-6.

[64] Wong, K. Y. and Spetsakis, M. E. Motion segmentation by EM clustering of good features. In *Proc. CVPR Workshop*, p. 166, June 2004.

[65] Du, W. and Piater, J. H. Tracking by Cluster Analysis of Feature Points and Multiple Particle Filters. In Wang, P. et al., editors, *Pattern Recognition and Image Analysis, 3rd Int. Conf. on Advances in Pattern Recognition, ICAPR, Proc., Part II*, volume 3687 of *LNCS*, pp. 701–710. Springer, 2005.

[66] Saunier, N. and Sayed, T. A feature-based tracking algorithm for vehicles in

intersections. In *3rd Canadian Conf. on Computer and Robot Vision (CRV)*, p. 59, June 2006. doi: 10.1109/CRV.2006.3.

[67] Dimitriou, N., Stavropoulos, G., Moustakas, K., and Tzovaras, D. Multiple object tracking based on motion segmentation of point trajectories. In *13th Int. Conf. on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 200–206. IEEE, August 2016. doi: 10.1109/AVSS.2016.7738057.

[68] Buddubariki, V., Tulluri, S. G., and Mukherjee, S. Multiple object tracking by improved klt tracker over surf features. In *5th National Conf. on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*, 2015.

[69] Bilinski, P., Bremond, F., and Kaaniche, M. B. Multiple object tracking with occlusions using hog descriptors and multi resolution images. In *3rd Int. Conf. on Imaging for Crime Detection and Prevention (ICDP)*, December 2009.

[70] Yu, J., Farin, D., and Loos, H. S. Multi-cue based visual tracking in clutter scenes with occlusions. In *6th Int. Conf. on Advanced Video and Signal Based Surveillance*, p. 158. IEEE, September 2009. doi: 10.1109/AVSS.2009.56.

[71] Byeon, M., Chang, H. J., and Choi, J. Y. Hierarchical feature grouping for multiple object segmentation and tracking. In *Proc. 27th Conf. on Image and Vision Computing*, pp. 452–457. ACM, 2012. doi: 10.1145/2425836.2425922.

[72] Kim, Z. Real time object tracking based on dynamic feature grouping with background subtraction. In *Proc. CVPR*. IEEE, June 2008.

[73] Wang, J. Y. A. and Adelson, E. H. Layered representation for motion analysis. In *Proc. CVPR*, pp. 361–366, June 1993. doi: 10.1109/CVPR.1993.341105.

[74] Fradet, M., Robert, P., and Pérez, P. Clustering Point Trajectories with Various Life-Spans. In *Conf. for Visual Media Production*, pp. 7–14, November 2009.

[75] Zeppelzauer, M., Zaharieva, M., Mitrovic, D., and Breiteneder, C. A novel trajectory clustering approach for motion segmentation. In *Advances in Multimedia Modeling: 16th Int. Multimedia Modeling Conf. (MMM)*, pp. 433–443. Springer, 2010. ISBN 978-3-642-11301-7. doi: 10.1007/978-3-642-11301-7_44.

[76] Nebehay, G. and Pflugfelder, R. Clustering of Static-Adaptive correspondences for deformable object tracking. In *Proc. CVPR*. IEEE, June 2015.

[77] He, S., Yang, Q., Lau, R. W. H., Wang, J., and Yang, M. H. Visual tracking via locality sensitive histograms. In *Proc. CVPR*, pp. 2427–2434, June 2013.

[78] Schreer, O. and Masneri, S. Automatic video analysis for annotation of human body motion in humanities research. In *9th Language Resources and Evaluation Conf., Int. Workshop on Multimodal Corpora*, 2014.

[79] Crasborn, O. and Sloetjes, H. Enhanced ELAN functionality for sign language corpora. In *Proc. 6th Int. Conf. on Language Resources and Evaluation (LREC)*. ELRA, 2008.

[80] Cox, I. J. and Hingorani, S. L. An Efficient Implementation of Reid's Multiple Hypothesis Tracking Algorithm and its Evaluation for the Purpose of Visual Tracking. *IEEE TPAMI*, 18(2):138–150, 1996. ISSN 0162-8828.

[81] Kalal, Z., Mikolajczyk, K., and Matas, J. Tracking-Learning-Detection. *IEEE TPAMI*, 34(7):1409–1422, July 2012. ISSN 0162-8828.

[82] Henriques, J. F., Caseiro, R., Martins, P., and Batista, J. Exploiting the Circulant

Structure of Tracking-by-Detection with Kernels. In *Proc. ECCV*, pp. 702–715. Springer, 2012. ISBN 978-3-642-33765-9.

[83] Danelljan, M., Khan, F. S., Felsberg, M., and v. d. Weijer, J. Adaptive color attributes for real-time visual tracking. In *Proc. CVPR*, pp. 1090–1097. IEEE, June 2014. doi: 10.1109/CVPR.2014.143.

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

[85] Held, D., Thrun, S., and Savarese, S. Learning to track at 100 fps with deep regression networks. In *Proc. ECCV*, 2016.

[86] Berman, Z. Outliers rejection in kalman filtering - some new observations. In *2014 IEEE/ION Position, Location and Navigation Symposium - PLANS 2014*, pp. 1008–1013, May 2014. doi: 10.1109/PLANS.2014.6851466.

[87] Park, D., Zitnick, C. L., Ramanan, D., and Dollár, P. Exploring weak stabilization for motion feature extraction. In *Proc. CVPR*, pp. 2882–2889. IEEE, June 2013.

[88] De Schutter, J., De Geeter, J., Lefebvre, T., and Bruyninckx, H. Kalman filters: A tutorial. 1999. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.28.7405.

[89] Bradski, G. R. and Kaehler, A. *Learning OpenCV - Computer Vision with the OpenCV Library*. O'Reilly Media, Inc., first edition, 2008. ISBN 978-0596-51613-0.

[90] Gamse, S., Nobakht-Ersi, F., and Sharifi, M. A. Statistical process control of a kalman filter model. *Sensors*, 14(10), 2014. doi: 10.3390/s141018053.

[91] Coraluppi, S. Fundamentals and advances in multiple-hypothesis tracking. Technical report STO-EN-IST-134, NATO, 2015.

[92] Bar-Shalom, Y. and Blair, W. D. *Multitarget-Multisensor Tracking: Applications and Advances*. Number v.3 in Artech House Radar Library. Artech House, 2000.

[93] Newman, A. J. and Mitzel, G. E. Upstream data fusion: History, technical overview, and applications to critical challenges. Technical Report 31 (3), Johns Hopkins APL Technical Digest, 2013.

[94] Mallick, M., Coraluppi, S., and Carthel, C. Multitarget Tracking Using Multiple Hypothesis Tracking. In *Integrated Tracking, Classification, and Sensor Management*, pp. 163–203. John Wiley & Sons, Inc., 2013. ISBN 978-11184-5055-0.

[95] Amditis, A., Thomaidis, G., Maroudis, P., Lytrivis, P., and Karaseitanidis, G. Multiple Hypothesis Tracking Implementation. In Rodriguez, J. A. M., editor, *Laser Scanner Technology*, chapter 10, pp. 199–220. InTech, 2012.

[96] Reid, D. B. An algorithm for tracking multiple targets. *IEEE Trans. Autom. Control*, 24(6):843–854, 1979. ISSN 0018-9286.

[97] Arras, K. O., Grzonka, S., Luber, M., and Burgard, W. Efficient People Tracking in Laser Range Data using a Multi-Hypothesis Leg-Tracker with Adaptive Occlusion Probabilities. In *Int. Conf. on Robotics and Automation, Pasadena, CA*, pp. 1710–1715. IEEE, 2008. ISBN 9781424416479.

[98] Coraluppi, S. and Carthel, C. Modified scoring in multiple-hypothesis tracking. *J. Adv. Inf. Fusion*, 7:153–164, 2012.

[99] Coraluppi, S. Multiple-hypothesis and graph-based approaches to multi-target

tracking. In *Advanced Algorithms for Effectively Fusing Hard and Soft Information*, number STO-EN-IST-155 AC/323(IST-155)TP/737 in STO Educational Notes. NATO Science and Technology Organization, 2016.

[100] Bar-Shalom, Y., Daum, F., and Huang, J. The Probabilistic Data Association Filter - Estimation in the Presence of Measurement Origin Uncertainty. *IEEE Control Syst. Mag.*, pp. 82–100, December 2009.

[101] Vo, B.-n., Mallick, M., Bar-shalom, Y., Coraluppi, S., Osborne, R., Mahler, R., Vo, B.-t., and Webster, J. G. Multitarget Tracking. In *Wiley Encyclopedia of Electrical and Electronics Engineering*, chapter 1. John Wiley & Sons, Inc., 2015. ISBN 978-04713-4608-1. doi: 10.1002/047134608X.W8275.

[102] Luber, M., Tipaldi, G. D., and Arras, K. O. Place-dependent people tracking. *Int. Journal of Robotics Research*, 30(3), March 2011.

[103] Wyffels, K. and Campbell, M. Negative observations for multiple hypothesis tracking of dynamic extended objects. In *2014 American Control Conference*, pp. 642–647, June 2014. doi: 10.1109/ACC.2014.6859041.

[104] Zúñiga, M. D., Brémond, F., and Thonnat, M. Real-time reliability measure-driven multi-hypothesis tracking using 2d and 3d features. *EURASIP J. on Advances in Signal Processing*, 2011(1):142, December 2011. ISSN 1687-6180.

[105] Ying, L., Xu, C., and Guo, W. Extended MHT algorithm for multiple object tracking. *Proc. 4th Int. Conf. on Internet Multimedia Computing and Service (ICIMCS)*, pp. 75–79, 2012.

[106] Manafifard, M., Ebadi, H., and Moghaddam, H. A. Appearance-based multiple hypothesis tracking: Application to soccer broadcast videos analysis. *Signal Processing: Image Communication*, 55:157–170, 2017. ISSN 0923-5965.

[107] Antunes, D. M., Figueira, D., Matos, D. M., Bernardino, A., and Gaspar, J. Multiple hypothesis tracking in camera networks. In *Proc. ICCV Workshops*, pp. 367–374, November 2011. doi: 10.1109/ICCVW.2011.6130265.

[108] McAnanama, J. and Kirubarajan, T. A multiple hypothesis tracker with interacting feature extraction. *Signal Processing*, 92:2962–2974, 2012.

[109] Kim, C., Li, F., Ciptadi, A., and Rehg, J. M. Multiple hypothesis tracking revisited. In *Proc. ICCV*, pp. 4696–4704. IEEE, 2015. ISBN 978-1-4673-8391-2.

[110] Lau, B., Arras, K. O., and Burgard, W. Multi-model hypothesis group tracking and group size estimation. *Int. Journal of Social Robotics*, 2(1):19–30, 2009. ISSN 1875-4791. doi: 10.1007/s12369-009-0036-0.

[111] Coraluppi, S. and Carthel, C. Multi-Stage Multiple-Hypothesis Tracking. *Journal of Advances in Information Fusion*, 6(1):57–68, 2011.

[112] Wigren, T. Target-type probability combining algorithms for multisensor tracking. In *Proc. SPIE*, volume 4380, pp. 4380–4397, 2001. doi: 10.1117/12.436991.

[113] Pannetier, B. and Dezert, J. Track segment association with classification information. In *Workshop on Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pp. 60–65, September 2012. doi: 10.1109/SDF.2012.6327909.

[114] Bendich, P., Chin, S. P., Clark, J., Desena, J., Harer, J., Munch, E., Newman, A., Porter, D., Rouse, D., Strawn, N., and Watkins, A. Topological and statistical behavior classifiers for tracking applications. *IEEE Trans. on Aerospace and*

Electronic Systems, 52(6):2644–2661, 2016. ISSN 0018-9251.

[115] Blasch, E. Modeling intent for a target tracking and identification scenario. In *Proc. SPIE*, volume 5428, April 2004.

[116] Lancaster, J. and Blackman, S. Joint IMM/MHT Tracking and Identification for Multi-Sensor Ground Target Tracking. In *2006 9th Int. Conf. on Information Fusion*, July 2006. doi: 10.1109/ICIF.2006.301798.