SEVENTH FRAMEWORK PROGRAMME

Networked Media

Specific Targeted Research Project

SMART

(FP7-287583)

Search engine for MultimediA environment generated contenT

D3.2.1 Video Signal Processing Prototypes

Due date of deliverable: 30-04-2013

Actual submission date: 10-10-2013

Start date of project: 01-11-2011 Duration: 36 months

Summary of the document

Code:	**D3.2.1 Video Signal Processing Prototypes**
Last modification: **State:**	2013-10-08 Final
Participant Partner(s): **Author(s):** **Fragment:**	AIT Aristodemos Pnevmatikakis No
Audience:	☐ public ☒ restricted ☐ internal
Abstract:	*This document describes the video processing algorithms and the resulting systems employed in SMART for metadata extraction.*
Keywords:	• *Video signal processing* • *Particle filters* • *Colour matching* • *Foreground segmentation* • *Face detection* • *Face tracking* • *Motion analysis* • *Crowd analysis*
References:	See end of document

Table of Contents

1 Executive Summary

1.1 Scope

The SMART system is about offering users information fused from the physical world of sensors and the virtual world of social networks and Linked Data. The information regarding the physical world stemming from different locations is collected by perceptual algorithms operating on sensor outputs. This document is about the visual processing perceptual components of SMART.

This document will be accompanied by a second version, due on end of April 2014. All functionality is already described in the current version. The future one will include more testing and fine-tuning on actual SMART visual data, as explained throughout the document.

1.2 Audience

The algorithms generating the visual metadata in a SMART edge node is of interest to a number of stakeholders including:

- **Developers of the SMART open source project**: SMART open source developers will understand the type of visual processing implemented. Together with the open source versions of the systems presented here, already made available to the community in the first SMART software release, developers will have at hand both the full algorithms of the proprietary systems and the software implementations of their simpler versions. Together the two can be of benefit to the developers knowledgeable in video signal processing.
- **SMART project members (notably members working in WP4 but also in WP5 and WP6 of the project)**: The present document explains how all the metadata presented in D4.1 “SMART Distributed Knowledge Base & Open Linked Data Mechanisms” are generated. Understanding both documents will allow for project partners to comment on other needs that might arise from the examples in these two documents.

1.3 Summary

Video signal processing in SMART aims at analysing crowds in outdoor scenes and individuals in indoors ones. Crowd analysis regards people as foreground objects and estimates how densely they occupy the scene at hand, how they move and what colours they wear. Individuals are analysed by tracking their faces around the scene. Both algorithm suites are detailed in the following chapters.

1.4 Structure

The document is structured as follows:

- Chapter 2 provides an introduction to the near and far field processing components of SMART.
- Chapter 3 discusses crowd analysis. The introduction is followed by a detailed account of the involved algorithms and their parameters. The effect of these parameters is quantified using the introduced F1-score as a metric.
- Chapter 4 describes face tracking. Again the introduction is followed by the theoretical background necessary for understanding particle filtering. The performance is evaluated next, with a discussion of the different parameters and algorithmic choices possible.
- Chapter 5 concludes the deliverable and identifies what is to be expected by its next version.

2 Introduction

SMART cameras observe people in cities. There are two distinct modes of observation, dictated both by legal and technological constraints.

The legal constraint has to do with the outdoor versus the indoor use of computer vision systems. Although there are some exceptions, in Europe privately owned outdoors computer vision systems are not allowed to observe the individual person[1], while indoors ones are. Outdoors deployments of computer vision systems are only allowed to view the scene from a distance, so that each person occupies only a handful of pixels, being unrecognisable.

The technological constraint has to do with the nature of the scene. Monitoring crowded spaces, where each individual is sensed (i.e. initialised and subsequently tracked as an individual entity) is still not practical due to initialisation, occlusion and clutter problems. Such scenes result e.g. at the waiting area of an airport, when a flight passenger stream emerges or in a crowded square. Normal scenes on the other hand allow for person tracking algorithms. Such are the scenes where people only occasionally do not appear as visually distinct.

Far-field sensing of people and vehicles is applied to outdoors scenes and crowded indoors scenes, and utilises the crowd analysis algorithms presented in chapter 3. Near-field sensing is applied to normal indoors scenes, and utilises the face tracking algorithms presented in chapter 4.

2.1 Video processing within SMART architecture

Video processing components can be found in the SMART edge nodes. They process signals from the cameras and derive metadata describing crowds and people. To do so, every such component runs a video processing algorithm to get some results that are supposed to populate feeds in the edge node database. Feed population demands that the results are formatted based on the description of the feeds.

The video processing algorithms must be written in C or C++ because they are quite processing-intensive. They employ third party libraries for basic signal processing blocks. The most widespread such library (and the one we use) is OpenCV2. Programming the video processing components in C makes communication with the edge node database via the feeds a bit more challenging, since in C (unlike for example Java) there is no native way to format the derived metadata in XML/JSON and transfer them to the database. Instead custom functions for metadata formatting need to be built and the transfer is based on the third party library cURL3. The usage of both these third party libraries for simple video processing and feed population is detailed in the tutorial simple_camera perceptual component found in the SampleClients\C_PP directory of our open source release and documented in our Trac at: http://opensoftware.smartfp7.eu/projects/smart/wiki/simple_camera

2.2 SMART video systems: public and proprietary

This document describes the proprietary algorithms and systems for video processing in SMART edge nodes. Only the edge nodes operated by consortium members are expected to have these running and providing metadata.

The various edge nodes processing video streams that are expected to be operated by the community will not have these systems available. Instead, we are providing open source systems utilizing the baseline algorithms discussed in this document. These versions can be found under the SampleClients\C_PP directory of our open source release and are documented in our Trac at:

[1] According to both the Spanish and the Greek Data Protection Agencies (DPA), the combination of camera resolution and minimum person distance should be such that the person is not recognizable by a human observer. Our crowd analysis system adheres to this rule.

[2] http://opencv.org/

[3] http://curl.haxx.se/

http://opensoftware.smartfp7.eu/projects/smart/wiki/PerceptualComponents. At the final section of the next two chapters the current status and the future plans for the open-source versions of the systems are outlined. In addition, the community can use this document to understand the proprietary algorithms and then build their own versions of the systems.

3 Crowd analysis

3.1 Introduction

When the cameras are observing spaces where people are expected to form crowds, either indoors (shopping malls) or outdoors (squares, parks, or shopping streets), the SMART systems are constrained not to observe the individual, but rather the crowd.

Crowds in SMART are modelled as groups of foreground pixels. Due to the far field nature of the observation, it is impossible to tell the individual apart in these pixel groups. Instead we employ an algorithm to extract the foreground pixels, and a set of other algorithms to analyse them yielding useful metadata for the various SMART applications.

3.2 Adaptive foreground segmentation

The chosen foreground segmentation algorithm is a variant of Stauffer's adaptive foreground estimation algorithm [Stauffer00]. According to this, a model is built for every pixel in the frame. Gaussian Mixture Models (GMM) comprising n_{GMM} Gaussians each model the different colours every pixel can receive in a video sequence. We work on the YCbCr colour-space, assuming the three components independent for simplicity. The weight of the Gaussians in the mixture is proportional to the time a particular Gaussian models best the colour of the pixel. By manipulating these weights we estimate the foreground patches in the image and we reconstruct a background one. In the following subsections the algorithm is detailed.

3.2.1 Initialising Gaussians

The first frame of the video is used to initialize the first Gaussian for every pixel. The mean value of the Gaussian is the vector describing the colour triplet of the pixel. The variance is initialized to some moderate initial value σ_{init}^2 (the same for every colour component), to allow colour triples with subtle differences in same pixel but future frames to still be modelled by the Gaussian.

Gaussians are also initialized when a colour triple does not match the models for the pixel. In this case the next empty slot of the model is used. If the number of Gaussians in the model is already reached, then the newest Gaussian (the one with the smallest weight) is deleted and the new Gaussian is initialised in its place.

3.2.2 Matching and updating Gaussians

For every new frame, the colour triples of every pixel are checked against the GMM of that pixel for possible matching. For this the Mahalanobis distance between every one of the n_{GMM} Gaussians in the model and the triple is calculated and compared to a threshold value d_{GMM}. The Gaussian with the smallest distance under d_{GMM} is selected as the matching one (the one describing the new triple well enough and certainly better than any other in the GMM) and is updated.

Gaussian updating involves the increase of its weight based on the current learning rate of the pixel, and the decrease of all other weights in the pixel GMM. The learning rate $a(x,y,t)$ depends on the pixel coordinates (x,y) and the frame time t. The weight update for a matched Gaussian is then:

$$w(x,y,t) = [1 - a(x,y,t)]\, w(x,y,t-1) + a(x,y,t) \quad (1)$$

and for a non-matched Gaussian is:

$$w(x,y,t) = [1 - a(x,y,t)]\, w(x,y,t-1) \quad (2)$$

After weight update the Gaussians in every pixel GMM are sorted (in descending order) based on their weights. The weights are then re-normalised to unity.

The Gaussian mean and variances are also updated. Mean update ensures that Gaussians are following slow changes to the appearance of the pixel. Variance update leads to narrow models for triples that do not vary, while allowing wider models for triples that vary a lot but in small steps. The variances are not allowed to drop below a threshold σ_{min}^2.

The learning rate $a(x,y,t)$ is adapted per pixel and time step [Pnevmatikakis06]: It is increased to learn flicker faster in the background, while it is decreased to protect small immobile foreground patches from being learnt too fast. Flicker adaptation is straightforward: very small foreground patches at pixels (x,y) are considered flicker, in which case the rate is scaled by $a_f > 1$:

$$a(x,y,t) = a_f a(x,y,t-1) \tag{3}$$

Adaptation for target protection on the other hand is only possible if the system has a tracker that validates the foreground patches as targets and provides their speed. In that case the rate is scaled by a factor $a_p(s,v) < 1$ that depends on the target's size s and speed v.

3.2.3 Merging Gaussians

Since the number of Gaussians in the per pixel GMMs is fixed to n_{GMM}, we add a dynamic modelling capability by merging Gaussians that although at initialisation were quite apart, they have been drifting closer in the colour-space via the Gaussian adaptation process. Gaussians whose means have Euclidian distance less than a threshold d_{merge} are merged together. The new Gaussian has a weight equal to the sum of the weights of the two merged Gaussians, while it has mean and variance the weighted average of the two.

3.2.4 Foreground from the Gaussian mixture pixel model

Given the current GMM for all the pixels, the Pixel Persistence Map (PPM) I_{PPM} can be built, in which every pixel is represented by the weight of the Gaussian from its GMM that best describes its current colour triple. Regions of the map with large values correspond to pixels that have colours that appear there for a long time, hence belong to the background. On the contrary, regions with small values correspond to pixels that have colours that appear there for a short time, hence belong to the foreground. The unfiltered foreground pixels are those with accumulated model weights for the Gaussians with larger weights than the matching one that is above a threshold τ. These foreground pixels are subjected to shadow removal [Xu05] and morphological clean-up to obtain the foreground image I_{frg}.

The shadow removal employed compares the unfiltered foreground pixel to the background pixel at the same location. The background image is estimated as the current frame pixels for the locations the algorithm labels as background, and the mean of the Gaussian having the largest weight for those locations the algorithm labels foreground. Shadow removal is controlled by two tolerances: The tolerance to brightness distortion BD and to colour distortion CD. Unfiltered foreground pixels that exhibit too large brightness distortion, i.e. discrepancy from the background image luminance, are checked for colour distortion, i.e. discrepancy from the background image chrominance. Large BD and CD tolerances lead to fewer pixels being labelled as shadow.

3.3 Crowd analysis

In SMART we currently perform static and dynamic crowd analysis based on the foreground image I_{frg} estimated by the algorithm discussed in section 3.2. Static analysis provides the crowd density and the primary crowd colours, where dynamic analysis is about crowd motion directions.

In camera views spanning wide areas it is interesting to split the analysis into regions of interest. In the example of Figure 1, there are two such regions: The Santander City Hall square (Plaza Ayuntamiento) at the lower part of the frame, and the street (Calle de Jesus del Monasterio) at the upper part of the frame.

The regions of interest are further divided into processing zones for two reasons:

- The regions of interest can have irregular shapes, as shown for the Santander City Hall square region in Figure 1. They are approximated by a number of rectangular processing zones.
- There is no need to process each area in a zone at the same level of detail: Areas closer to the camera can be processed after their pixels are decimated. In the Santander City Hall square region there are two such processing zones defined, the near one being processed at a decimation factor of 4, and the far one being processed at a decimation factor of 2. The street region is described by only one processing zone, where all the pixels are processed (decimation factor of 1).

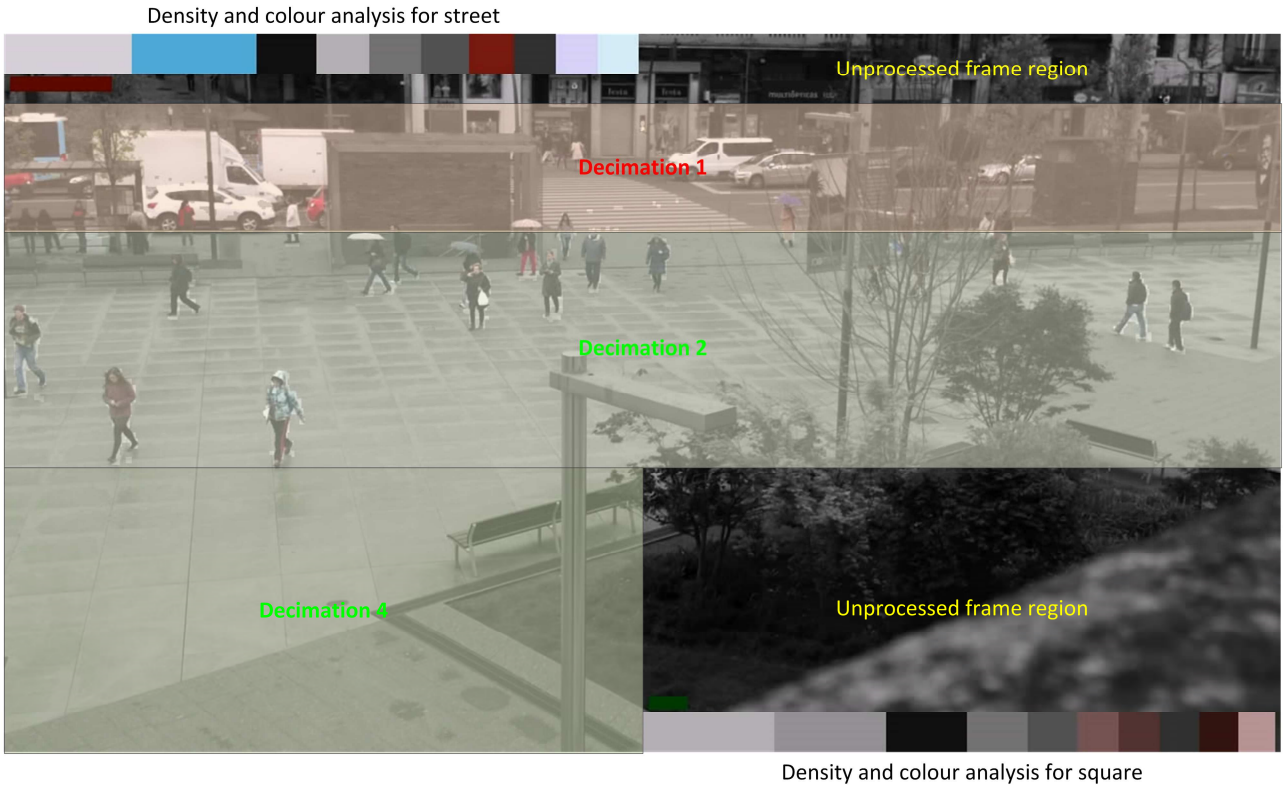

Figure 1: **Example of static crowd analysis in two regions (street and square). The second region is non-rectangular and is split into two rectangular processing zones utilising different decimation factors (as one zone is nearer and the other is further away from the camera). Foreground pixels are coloured while background are grey with reduced brightness. The size of the rectangles in the colour stripes indicates the frequency of appearance of the given colour in the foreground. The green and red narrow bars indicate the density of the foreground in the two zones. The video of the system can be viewed at: http://www.youtube.com/watch?v=q4HJtkN8Wr8.**

3.3.1 Crowd density

Crowd density is estimated from the foreground mask, by counting the pixels in the different regions of interest. The active pixels in different rows in the mask are weighted differently, as the distance from the camera decreases for rows closer to the top of the frame and hence the size of the objects' projection onto the camera frame reduces.

An example of crowd density estimation is shown in Figure 2.

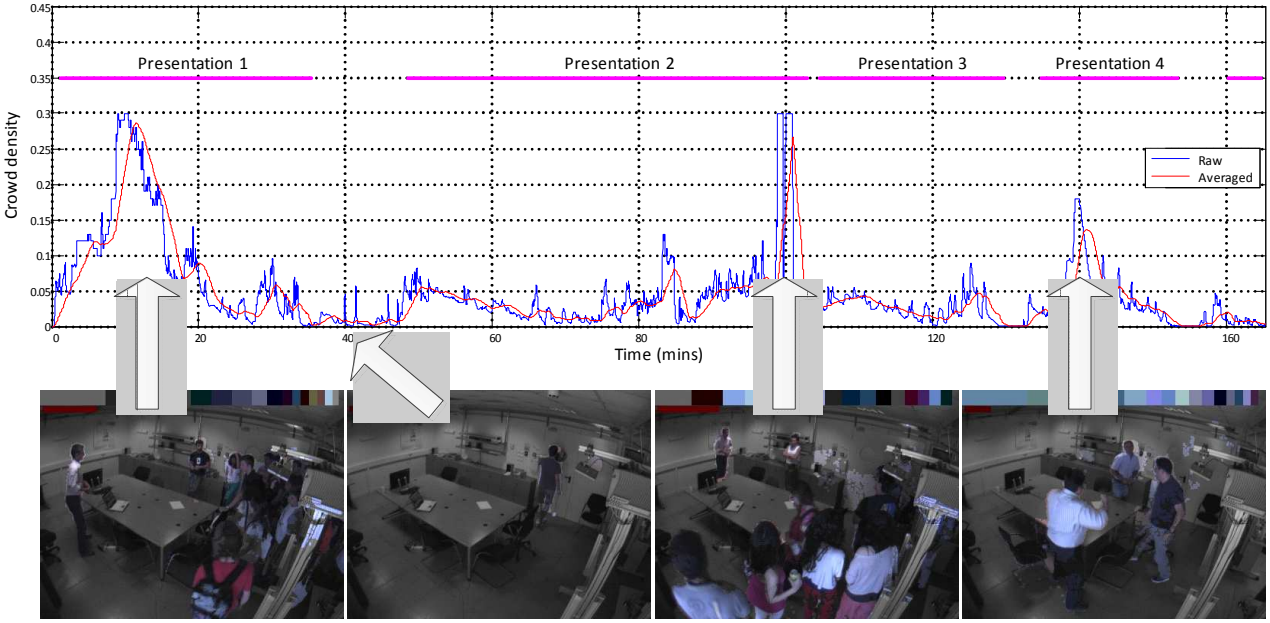

Figure 2: Crowd density estimation (top) during some activities at the AIT SmartLab (four sessions of presentations). Example processed frames are also shown (bottom) displaying the foreground areas and the prominent colours.

3.3.2 Crowd colours

For some (fashion-related) applications of the live news use case it is useful to know the colours people are wearing. For this we perform a colour histogram analysis of the foreground pixels and we report the most prominent colours and their frequency of appearance.

The colour histogram is updated periodically using some memory. Between updates, a temporary colour histogram is built for all foreground pixels in all the frames in that interval. The reported colour histogram is updated using the temporary one. After the update the temporary is reset to zero. The frequency of the update is unreasonable frequent (every 1,000 frames) in our demo video (http://www.youtube.com/watch?v=q4HJtkN8Wr8); it should be closer to daily updates for a real deployment.

Another example of the colour analysis is given in Figure 2.

Currently colours are reported in the RGB colour-space, but we will be exploring other perceptually-optimal ones, as well as compensating for lighting intensity by more crudely quantising the luminance component.

3.3.3 Crowd motion

The motion of crowds is very important in security applications. There are some directions of motion that are important and the system reports if they are exercised. The directions of motion of interest in our example are shown in Figure 3.

Motion vectors are extracted using the traditional block matching approach for every 16x16 block that has significant number of foreground pixels and significant texture variation. The former guarantees that we are not looking for a match to a background image patch, while the latter that there is significant information to find a robust motion vector. This is important, as contrary to motion estimation for image representation in video compression, here our goal is not to find a very similar block but to understand how the block is actually moving. The extracted motion vectors are then compared to the direction of interest vectors by means of their internal product.

Figure 3: **Directions of motion of interest for the two regions. For the road we are interested in motion along its axis, while for the square we are interested in pedestrian crossing its length, or moving between the zebra crossing and its left or right sides. Any other direction of interest for an application can be defined.**

3.4 Results

The crowd analysis SMART system is intended primarily for outdoors use, even though it is being used indoors at the AIT edge node. Outdoors data collection is still a problem for the project. Although the legalities have been recently resolved, we do not yet have at hand the SMART outdoor development and testing videos. For this reason the system is only superficially tested on a video captured by an iPhone from a window of Santander's Town Hall. The sequence is about ten minutes long and spans quiet and busy intervals in the depicted street and square. To quantify the performance of the system and the effect of the various parameters introduced in Section 3.2, we need to annotate every foreground pixel in the sequence. Since the sequence is not the final one to be used for algorithmic development and evaluation, we only annotated some frame spanning 100 seconds, obtaining a total of 131k manually labelled foreground pixels. This amount of annotation is suitable for the initial tuning and performance evaluation of the algorithm. We will repeat this more rigorously in the second version of this deliverable, due on month 30.

In the rest of this section we will be evaluating the performance of our foreground estimation algorithm on a per pixel level. We begin by introducing our evaluation metrics and compare our system to the baseline implementations. We then we proceed in evaluating the effect of the various parameters introduced in Section 3.2.

3.4.1 Evaluation metrics and comparison to baseline

Foreground estimation is actually a two-class classification problem: every pixel is classified in the foreground or the background class. As such, we employ the established metrics of precision, recall, and their combination in the F1-score for a single-number performance indicator combining both precision and recall [F1score13]. Some definitions are due:

- True positives TP : The pixels that are classified as foreground and are indeed foreground.
- False positives FP : The pixels that are classified as foreground but are actually background. This is one possible error that system can make.
- False negatives FN : The pixels that are classified as background but are actually foreground, or the missing foreground pixels. This is the other possible error that system can make.

- Precision $prec$ (positive predictive value): The ratio of correctly classified foreground pixels over the total pixels classified as foreground, i.e. $prec = \frac{TP}{TP + FP}$.
- Recall rec (true positive rate or sensitivity): The ratio of correctly classified foreground pixels over the sum of the correctly classified and the missing foreground pixels, i.e. $rec = \frac{TP}{TP + FN}$.
- F1-score: Usually when trying to improve one of the precision or recall, the other worsens. It is easier to find a single figure of merit, and this is the F1-score, defined as $F1 = \frac{2\,prec \cdot rec}{prec + rec}$.

The baseline system against which we compare our algorithm is the plain Stauffer implementation, where there is no learning rate adaptation, neither Gaussian merging. On top of that we add our improvements. The resulting performance is shown in Table 1.

Table 1: Performance comparison of the various foreground estimation system versions and the baseline Stauffer implementation.

Additions to Stauffer			Precision	Recall	F1-score	Notes
Rate adaptation		Gaussian merging				
Condition	Flicker					
No	No	No	0.1855	0.8979	0.3074	Baseline
Yes	No	No	0.2263	0.8524	0.3577	
No	Yes	No	0.5252	0.8493	0.6490	
Yes	Yes	No	0.7140	0.7924	0.7512	Dec. 2012
No	No	Yes	0.6358	0.8243	0.7179	
Yes	No	Yes	0.7227	0.7972	0.7581	
No	Yes	Yes	0.7901	0.7706	0.7802	
Yes	Yes	Yes	0.8249	0.7640	0.7933	March 2013

The performance increase over the baseline is 158%, mainly achieved by reducing the false positives. The performance increase over the Dec. 2012 (1st year) system that featured only rate adaptation is 5.61%. The results in Table 1 indicate that either rate adaptation or Gaussian merging grossly improve performance, with the former being somewhat more effective. Also, the local rate adaptation for flicker pixels learning is more important than the global rate adaptation for change of imaging conditions learning.

3.4.2 Effect of Stauffer parameters

In this subsection we examine the effect of the parameters of the baseline Stauffer algorithm:

- Number of Gaussians n_{GMM}. The accurate modelling of the background requires some Gaussians, and so does the frequently changing foreground. Hence the performance is expected to increase as n_{GMM} increases, up to the point that the extra Gaussians do not matter anymore. Then only the processing requirements increase. The experiments depicted in Figure 4 actually show that the performance even drops when n_{GMM} is increased beyond need. This is due to Gaussians of small weights surviving for longer (since there is no need to remove them as more slots are available for new ones). So these Gaussians have the opportunity to receive

more weight and in extreme cases even enter the background model. Tuning n_{GMM} is of intermediate difficulty, since performance peaks at the optimum value rather sharply.

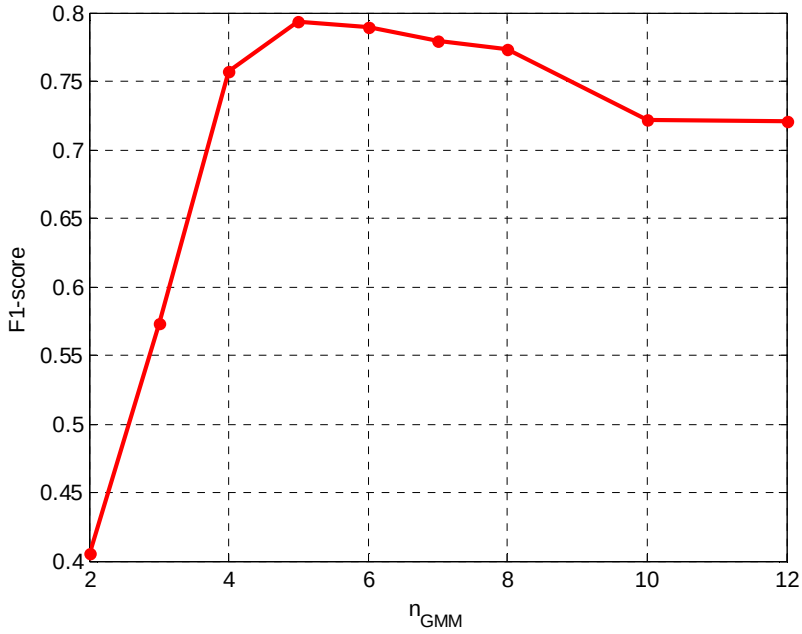

Figure 4: Effect of the number of Gaussians in adaptive foreground segmentation.

- Gaussian model variance (initial σ^2_{init} and minimum σ^2_{min}). The effect of the two variance parameters is presented in Figure 5 and Figure 6 respectively.

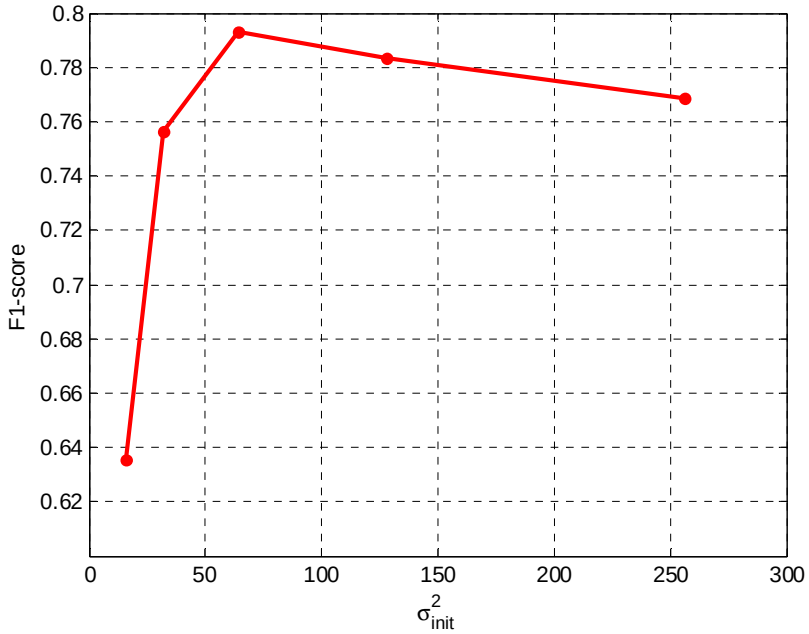

Figure 5: Effect of the initial variance of a Gaussian in adaptive foreground segmentation.

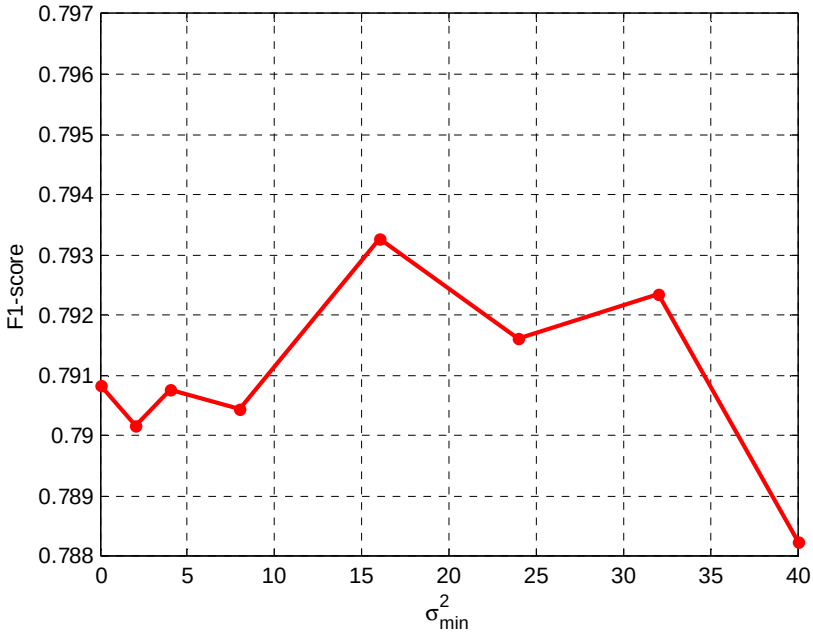

Figure 6: Effect of the minimum variance of a Gaussian in adaptive foreground segmentation.

The initial variance obviously is an important parameter, with performance peaking at $\sigma_{init}^2 = 64$. Tuning σ_{init}^2 is of intermediate difficulty since the range of near-optimum values is not wide.

The minimum variance on the other hand does not have a significant impact on performance, as is evident from the minor and non-monotonic variation of the F1-score of Figure 6. Nevertheless, we select the value $\sigma_{min}^2 = 16$ for which we measure a slight performance peak. Tuning σ_{min}^2 is very easy since performance is relatively constant as long as $\sigma_{min}^2 < 32$.

- Base learning rate a. This is the rate used when no rate adaptation is involved. The effect of varying the base rate is shown in Figure 7. Obviously this is a very important parameter, affecting performance in a smooth manner and peaking at $a = 2 \cdot 10^{-4}$. Although the effect of a on performance is strong, the performance peak is wide allowing for easy tuning of the learning rate.

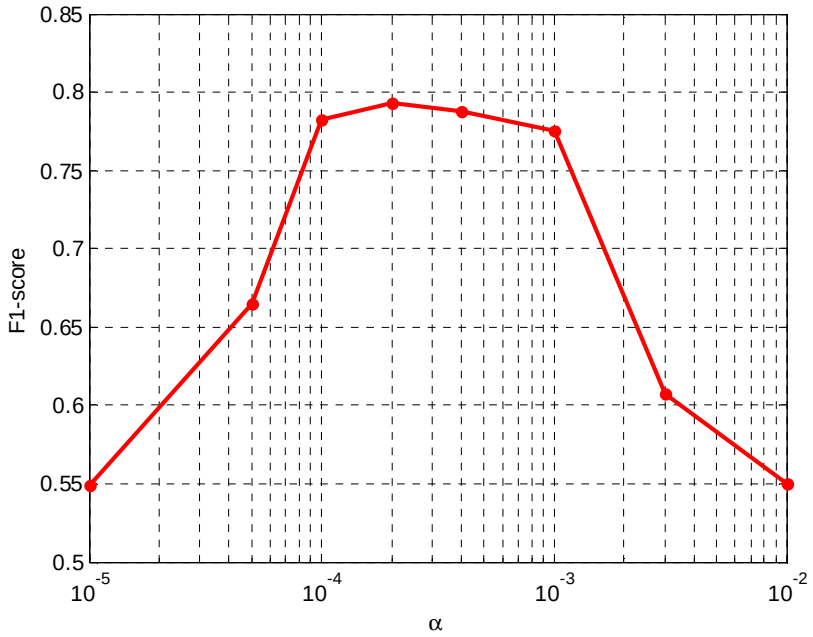

Figure 7: Effect of the base learning rate in adaptive foreground segmentation.

- Cumulative weight threshold τ. This parameter governs which Gaussians belong to the foreground and which to the background. There is only a narrow range of useful values for τ, as shown in Figure 8. Again, this is a parameter with smooth effect on performance, peaking at $\tau = 0.65$. Tuning τ is easy since the performance peak is wide.

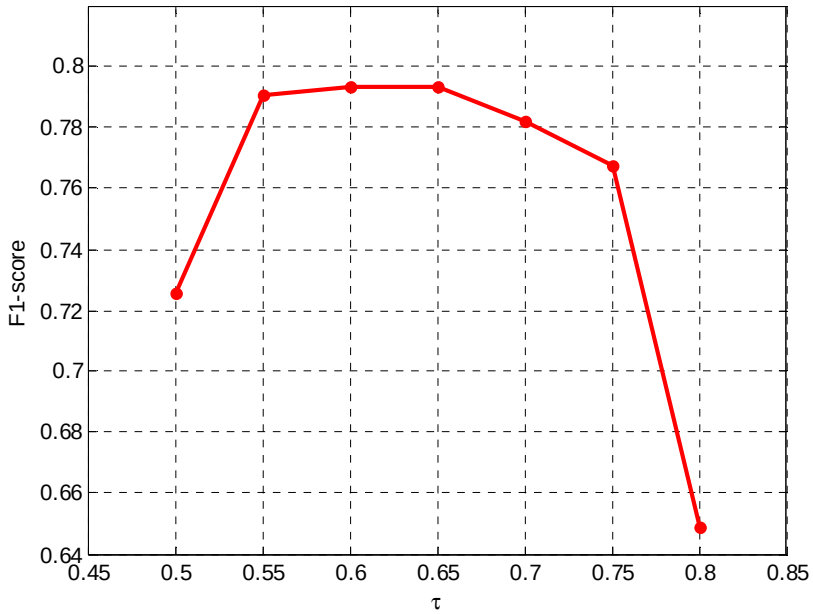

Figure 8: Effect of the cumulative weight threshold in adaptive foreground segmentation.

3.4.3 Effect of rate adaptation

In this subsection we examine the effect of two rate adaptation factors:

- The global rate adaptation factor after a camera or lighting condition change is detected. The learning rate is multiplied by that factor, temporarily increasing the learning rate throughout the frame. At each new frame the rate drops back by a factor of 2. There is a wide range of near optimum values for this factor, as shown in Figure 9. Hence tuning the global adaptation factor is easy.

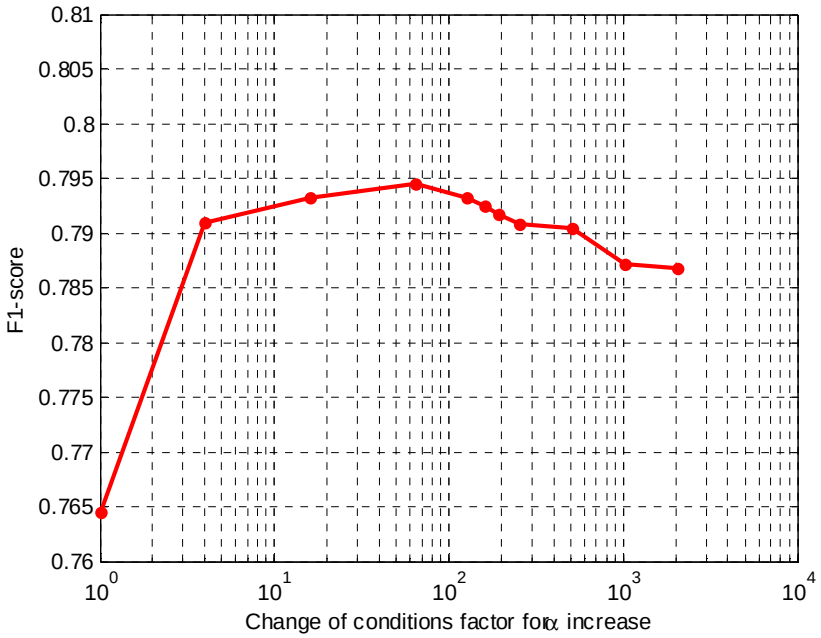

Figure 9: Effect of the global adaptation factor in adaptive foreground segmentation.

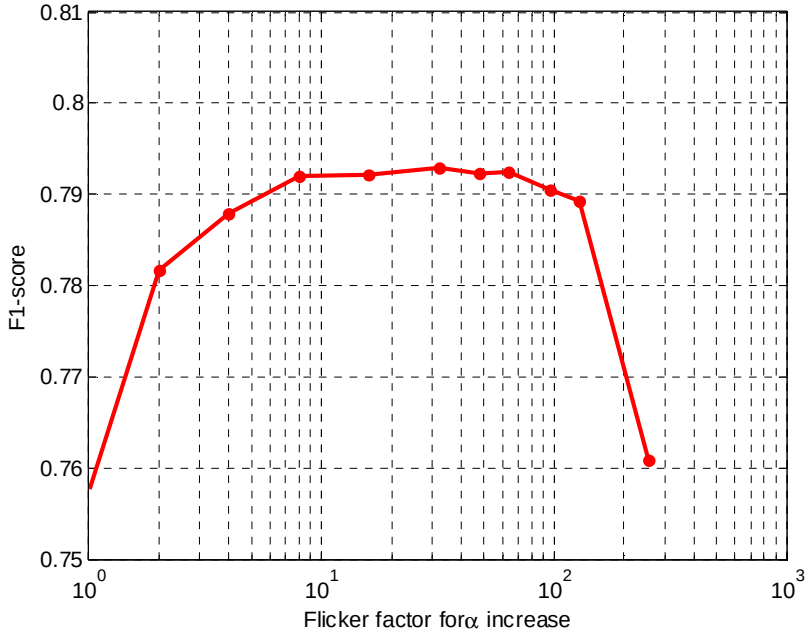

Figure 10: Effect of the local (flicker) adaptation factor in adaptive foreground segmentation.

- The local rate adaptation factor for every foreground pixel considered flicker. Flicker pixels are those detected as foreground but forming only very small blobs. The learning rate is locally increased by the multiplicative factor, resulting to the performance of Figure 10. The local rate factor is easily tuned since the performance peak is very wide.

3.4.4 Effect of Gaussian merging

Finally, in this subsection we examine the effect of the Gaussian merging distance threshold on performance. As shown in Figure 11, there is a rather narrow performance peak at a Euclidean distance between the two candidate Gaussians of 500. Hence the threshold is tuned with moderate difficulty.

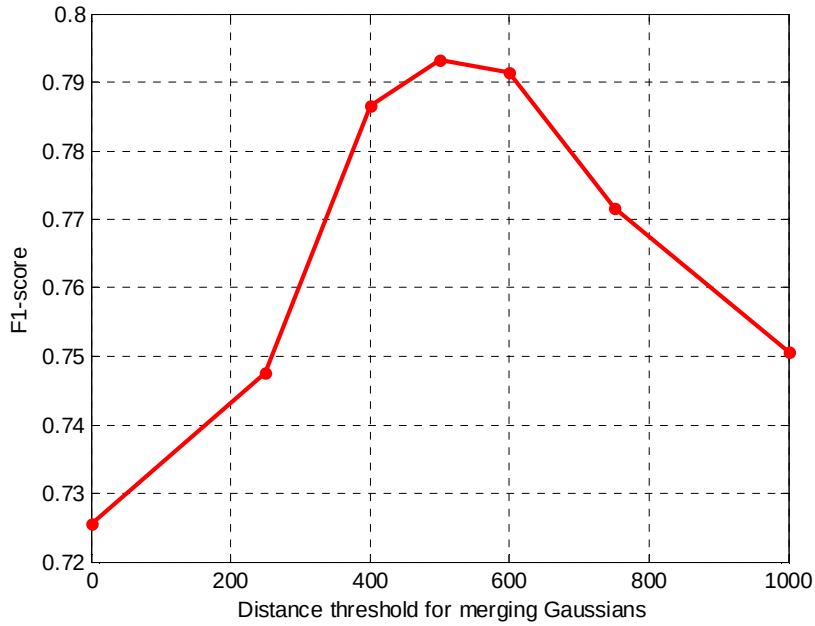

Figure 11: Effect of Gaussian merging threshold in adaptive foreground segmentation.

3.5 Open issue: Shadows

The biggest remaining detrimental factor of the foreground estimation system is shadows. An example is shown in Figure 12, where many of the shadow pixels are not actually removed. A better shadow detector will reduce false positives in foreground estimation, further improving precision. Our research in the next quarter will involve improving the shadow detector of the system.

We will attempt better shadow detection (or immunity) by a combination of:

- Better shadw detection algorithm
- Different Gaussian modelling of the luminance than the chrominance components (larger luminance variance)

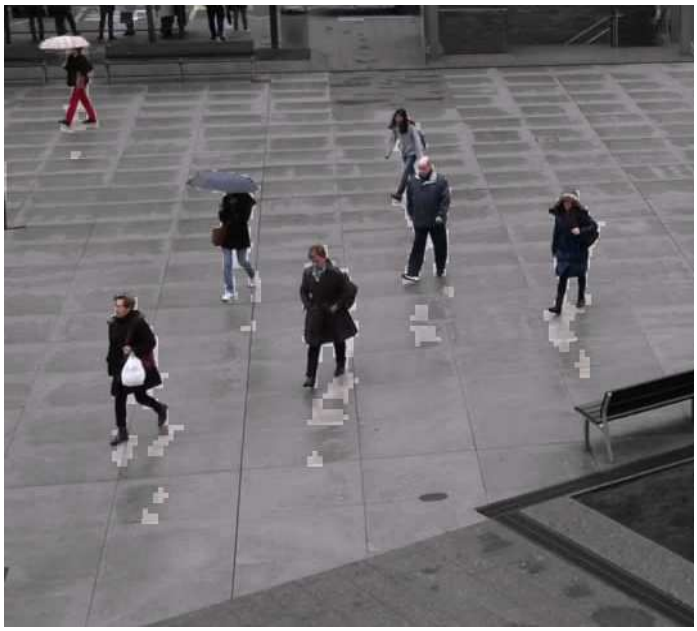

Figure 12: The current shadow detector only removes some of the shadow pixels.

3.6 Open-source crowd analysis system

The system described above remains proprietary to the SMART consortium. To facilitate video processing at edge nodes the community sets up, we have already distributed an open-source version of the system. This can be found under the SampleClients\C_PP directory of our open source release and is documented in our Trac at:

http://opensoftware.smartfp7.eu/projects/smart/wiki/simple_crowd_analysis

3.6.1 Current status

Currently the system only extracts visual density at the different regions specified manually by the user. It utilises the OpenCV implementation of the Gaussian Mixture-based Background/Foreground Segmentation Algorithm found in the BackgroundSubtractorMOG2 class [OpenCVMOG2].

3.6.2 Future plans

Our immediate plans foresee the inclusion of:

- Motion analysis (motion vector extraction, compliance to directions of interest and motion to or from a point).
- Colour analysis (prominent colours).
- Regions' and processing zones' definition tool (define rectangles for zones, associate a decimation factor to each and combine them into regions).

3.6.3 Limitations

We do not plan to include any of the following functionality in the open source system:

- Learning rate increase due to change of illumination conditions or camera movement.
- Learning rate increase for flicker reduction
- Learning rate decrease for blob protection
- Gaussian acceptance threshold adaptation to pixel intensity.

4 Face tracking

4.1 Introduction

Face tracking is very important for a series of visual processing systems that extract context from video signals and offer their users services. The tracks of faces are the most robust cues for the presence of humans in a monitored space, especially under clutter where the human bodies are no longer distinct [Bernardin09]. In such cases blobs do not yield human bodies, at least distinct ones, and initialisation must resort to some detection algorithm like the Boosted Cascades of Simple Features [Viola01] or the Histogram of Oriented Gradients [Ding09]. No matter the detection algorithm, faces are easier to initially detect in a video, since they flex much less than the body.

Once obtained by a tracker, face tracks form very important metadata in a variety of applications:

- Service provision in smart working [Waibel04, CHIL] or living [HERMES] spaces
- Security and sensitive infrastructure monitoring
- Enhancing viewers' experience of broadcasted material [Patrikakis10]

Although in some controlled environments face tracking yields robust results, in more challenging ones a tracking system encounters difficulties due to background clutter. The tracker needs face models that are robust enough to discriminate the faces from such clutter. These models are of the form of a measurement likelihood given the target state, can be derived from prior knowledge and are usually somehow updated to retain their discriminating ability over time.

4.1.1 Tracking background

A detector is the algorithm that searches for objects of interest in the frame available from the camera at a particular instance in time. For the SMART face tracking system, the objects are faces and the detector localises them in any frame.

A tracker on the other hand is the algorithm that propagates the knowledge about the location of the object of interest, termed a target, across time and space as more video frames become available. Both algorithms suffer from clutter: human faces are sensed in the presence of multiple competing irrelevant signals in the form of image patches resembling or occluding the faces. In this challenging environment a tracker needs to utilise all available information from the video sequence, utilising diverse measurement types. Note that clutter with target resemblance is grossly different when we measure different things. E.g., a skin-coloured wall will resemble a face a lot under colour measurements, but not at all under texture measurements.

All the relevant information about the target to be tracked is contained in its state vector $\mathbf{x}$. In its simplest form, the state comprises position variables but tracking is facilitated when more information is provided, forming a model of the target. Simple models to be included in the state can be the size of the target, its orientation, the situation it is in (e.g. facial expression, the person's identity) and its motion (velocity, acceleration, type of motion). More elaborate target models include specialised information like a colour model. Hence the state of a target can have a large number of dimensions, with elements that receive continuous, discrete or binary values.

Given the state vector at the previous time instant $\mathbf{x}_{n-1}$ and the sequence of measurement vectors $\mathbf{y}_{1:n}$ since time 1 up to the current time n, the tracker estimates the state at the current time instant $\mathbf{x}_n$. The estimation is usually done in two stages.

First, all knowledge of the motion habits of the target is utilised. This knowledge is encapsulated by the object model. This can be some deterministic kinematic model, such as the constant velocity model, or a probabilistic one, introducing the process noise $\mathbf{v}_n$ in the target dynamics, expressing uncertainty as to its exact nature. The (possibly time-varying) object model $\mathbf{f}_n$ shifts $\mathbf{x}_{n-1}$ into the one-step state prediction $\mathbf{x}_n$ that is based only on the knowledge of the motion habits:

$$\mathbf{x}_n = \mathbf{f}_n\left(\mathbf{x}_{n-1}, \mathbf{v}_n\right) \tag{4}$$

The second estimation step is the measurement update. The information of the pixels around the one-step state prediction is utilised to update the estimate using some video processing measurements. To measure around the one-step state prediction, a (possibly time-varying) probabilistic measurement model $\mathbf{h}_n$ is utilised that connects the current state $\mathbf{x}_n$ with the current measurement $\mathbf{y}_n$, introducing measurement noise $\mathbf{u}_n$ to account for the noisy measurement process:

$$\mathbf{y}_n = \mathbf{h}_n(\mathbf{x}_n, \mathbf{u}_n) \tag{5}$$

Due to the stochastic nature of the models, the current state $\mathbf{x}_n$ becomes a random variable. Stochastic tracking (or Bayesian filtering), is the process of estimating the posterior probability density function (PDF) of $\mathbf{x}_n$, given all the measurement history $\mathbf{y}_{1:n}$. In effect this is the conditional PDF of the state, conditioned upon the evidence, $p(\mathbf{x}_n|\mathbf{y}_{1:n})$.

Bayesian filtering can be achieved recursively by estimating the posterior $p(\mathbf{x}_n|\mathbf{y}_{1:n})$ at time n from the posterior $p(\mathbf{x}_{n-1}|\mathbf{y}_{1:n-1})$ at previous time $n-1$ and the current measurement $\mathbf{y}_n$. This is done in two steps. First the previous posterior is mapped into the one-step prediction density $p(\mathbf{x}_n|\mathbf{y}_{1:n-1})$ utilising all the available information about the current state $\mathbf{x}_n$, i.e. the previous state $\mathbf{x}_{n-1}$ and the sequence of past measurements $\mathbf{y}_{1:n-1}$. This is expressed as the conditional PDF $p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{y}_{1:n-1})$. Then:

$$p(\mathbf{x}_n|\mathbf{y}_{1:n-1}) = \int_{\mathbf{x}_{n-1}} p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{y}_{1:n-1})\, p(\mathbf{x}_{n-1}|\mathbf{y}_{1:n-1})\, d\mathbf{x}_{n-1} \tag{6}$$

where the integration is marginalisation across all possible previous states $\mathbf{x}_{n-1}$.

The posterior is then obtained by utilising the most recent measurement $\mathbf{y}_n$ and Bayes' rule in the measurement update step:

$$p(\mathbf{x}_n|\mathbf{y}_{1:n}) = \frac{p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{y}_{1:n-1})\, p(\mathbf{x}_n|\mathbf{y}_{1:n-1})}{p(\mathbf{y}_n|\mathbf{y}_{1:n-1})} \tag{7}$$

The posterior contains all the information necessary for estimating the target state. According to the Maximum-a-Posteriori (MAP) estimation, the estimated state is the one that maximises the posterior:

$$\mathbf{x}_n^{MAP} = \arg\max_{\mathbf{x}_n} \left(p(\mathbf{x}_n|\mathbf{y}_{1:n}) \right) \tag{8}$$

According to the Minimum Mean Square Error (MMSE) estimation, the estimated state is the expectation of $\mathbf{x}_n$ conditioned upon the evidence, i.e. the measurement history:

$$\mathbf{x}_n^{MMSE} = \int_{\mathbf{x}_n} \mathbf{x}_n\, p(\mathbf{x}_n|\mathbf{y}_{1:n})\, d\mathbf{x}_n \tag{9}$$

The conditioning upon the past measurement history both in the prediction (6) and in the measurement update (7) steps make recursive Bayesian filtering impossible. The conditioning on $\mathbf{y}_{1:n-1}$ needs to be eliminated from all terms but the given previous posterior; this is achieved by two simplifying assumptions for conditional independence:

- Given the previous state, the current state is independent of the past measurement history.
- Given the current state, the current measurement is independent of the past measurement history.

Then the prediction (6) and in the measurement update (7) steps are simplified [Talantzis12, section 2.4.1] to:

$$p\left(\mathbf{x}_n \left| \mathbf{y}_{1:n-1}\right.\right) = \int_{\mathbf{x}_{n-1}} p\left(\mathbf{x}_n \left| \mathbf{x}_{n-1}\right.\right) p\left(\mathbf{x}_{n-1} \left| \mathbf{y}_{1:n-1}\right.\right) d\mathbf{x}_{n-1} \tag{10}$$

and:

$$p\left(\mathbf{x}_n \left| \mathbf{y}_{1:n}\right.\right) \propto p\left(\mathbf{y}_n \left| \mathbf{x}_n\right.\right) p\left(\mathbf{x}_n \left| \mathbf{y}_{1:n-1}\right.\right) \tag{11}$$

Note that in this form of the recursive Bayesian estimation, the object and measurement models are involved in the form of the object $p\left(\mathbf{x}_n \left| \mathbf{x}_{n-1}\right.\right)$ and measurement $p\left(\mathbf{y}_n \left| \mathbf{x}_n\right.\right)$ likelihood functions.

Usage of the prediction (10) and in the measurement update (11) steps under Gaussianity and linearity assumptions yields the Kalman Filter [Talantzis12, section 2.4.2]. In the more general case of non-linear object and measurement models, their solution is done numerically using the Particle filter [Arulampalam02].

4.1.2 Particle filtering background

Particle filtering is based on two principles:

- The approximation of distributions with discrete particles: A target distribution $p(\mathbf{x})$ can be approximated by a set of N_p discrete samples $\mathbf{x}^{(i)}$ and their associated weights $w^{(i)}$. Jointly the samples and the weights form the particles, $\left\{\mathbf{x}^{(i)}, w^{(i)}\right\}_{i=1}^{N_p}$ [Talantzis12, section 2.4.3.1].
- Importance sampling: There are cases where the target distribution $p(\mathbf{x})$ can be evaluated for any $\mathbf{x}$, but samples cannot be drawn from it. Instead they can be drawn from a proposal distribution $q(\mathbf{x})$, yielding the set $\left\{\mathbf{x}^{(i)}, 1/N_p\right\}_{i=1}^{N_p}$. For the particles drawn from the proposal distribution to approximate the target one, the weights are calculated as $w^{(i)} \propto \frac{p(\mathbf{x}^{(i)})}{q(\mathbf{x}^{(i)})}$ [Talantzis12, section 2.4.3.2]

According to the particle filtering framework, a set of N_p particles $\left\{\mathbf{x}_{n-1}^{(i)}, w_{n-1}^{(i)}\right\}_{i=1}^{N_p}$ approximating the posterior $p\left(\mathbf{x}_{n-1} \left| \mathbf{y}_{n-1}\right.\right)$ at time $n-1$ is updated to $\left\{\mathbf{x}_n^{(i)}, w_n^{(i)}\right\}_{i=1}^{N_p}$ approximating the posterior $p\left(\mathbf{x}_n \left| \mathbf{y}_n\right.\right)$ at time n.

Following the mathematical derivation of [Talantzis12, section 2.5], the Sequential Importance Resampling particle filter recursion involves the following three steps:

Draw the updated particles $\mathbf{x}_n^{(i)}$ from the proposal distribution $q\left(\mathbf{x}_n \left| \mathbf{x}_{n-1}^{(i)}, \mathbf{y}_n\right.\right)$, after conditioning it upon the previous samples $\mathbf{x}_{n-1}^{(i)}$ and the current measurement $\mathbf{y}_n$. The most common choice for the proposal distribution is the object model itself.

Obtain the current weights $w_n^{(i)}$. To do so the measurement model $p\left(\mathbf{y}_n \left| \mathbf{x}_n\right.\right)$, the object model $p\left(\mathbf{x}_n \left| \mathbf{x}_{n-1}\right.\right)$ and the proposal distribution $q\left(\mathbf{x}_n \left| \mathbf{x}_{n-1}, \mathbf{y}_n\right.\right)$ are evaluated at the newly drawn samples $\mathbf{x}_n^{(i)}$, the previous ones $\mathbf{x}_{n-1}^{(i)}$ and the current measurement $\mathbf{y}_n$, as:

$$w_n^{(i)} \propto \frac{p\left(\mathbf{y}_n \middle| \mathbf{x}_n^{(i)}\right) p\left(\mathbf{x}_n^{(i)} \middle| \mathbf{x}_{n-1}^{(i)}\right)}{q\left(\mathbf{x}_n^{(i)} \middle| \mathbf{x}_{n-1}^{(i)}, \mathbf{y}_n\right)} \tag{12}$$

Resample the particles, a process that replaces the samples with very small weights by those with large weights.

The operation of the SIR particle filter is summarised in Figure 13.

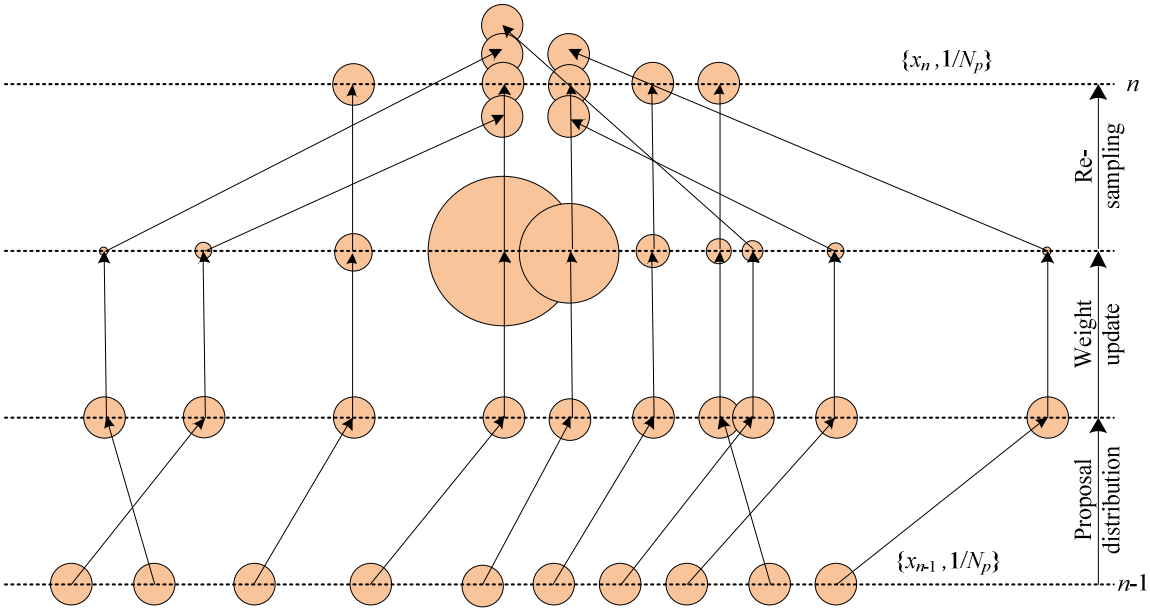

Figure 13: Representation of the three steps of the SIR particle filter recursion.

Any tracker based on particle filtering comprises:

- the object model propagating the prior state to the current time instance,
- the proposal distribution to draw the samples from, which is usually the same as the object model,
- the measurement likelihood for an observation given the state, and
- the target management for (re)initialization and termination.

The main innovation of the SMART face tracker is target modelling by three measurement likelihood functions suitable for face tracking. We then proceed to fuse the three measurement cues using an early fusion scheme into a particle filter tracker, suitable for our non-linear measurement models [Arulampalam02].

Secondary innovations of the system involve the measurement-assisted object model and the target management approach.

4.1.3 Face models

Face modelling approaches based on tracking by detection employ a discriminative classifier. Such a classifier can be trained in a batch mode prior to application in a tracking system. Typical examples include the Boosted Cascades of Simple Features [Viola01] or the Histogram of Oriented Gradients [Ding09] classifier. Adverse conditions result in bad framing of the target, which, when accumulated, lead to tracker drift and finally target loss. On the other hand the classifier can be initialised and subsequently updated. In this case the appearance model of the target is adaptive, with typical example being colour modelling [Jaffre03, Jones02]. Traditionally, adaptive modelling involves the use of heuristically-derived forgetting factors. Recently online multiple instance learning [Babenko09] overcame this problem by retraining a discriminative classifier.

Our approach to overcome the changes of the face and updating appearance models is to use multiple models. We employ:

- a model based on boosted cascades of simple features that drives face detection measurements and is not trained on the target,
- a colour model that is trained on the target and requires infrequent retraining, and
- a foreground model that is not trained at all as it only looks for foreground blobs that have background on their side and top.

These models are different in terms of persistence and discriminating ability. Face detection is very discriminative for frontal faces but is very sensitive to poses or occlusions. The discriminative power of colour models depends on the colours of the background and suffers from illumination changes and target pose changes. Finally, foreground segmentation distinguishes a moving face from the immobile background well, but is sensitive to camera motion, background changes and immobile targets. Although our standalone models suffer in persistence or discriminating ability, their combination with our proposed tracker yields robust performance.

4.2 Likelihood functions

In this section we derive the three measurement likelihoods to be used in our face tracker. Dropping the time dependence the likelihood functions are $p(\mathbf{y}|\mathbf{x})$, where the measurement $\mathbf{y}$ can is either of:

- $\mathbf{y}_{obj}$ for object presence measurements (face detection in our case),
- $\mathbf{y}_{frg}$ for foreground measurements, and
- $\mathbf{y}_{cm}$ for colour matching measurements.

4.2.1 Object presence

Object likelihood represents the certainty of object presence at the image patch $I_{\mathbf{x}}$ corresponding to the state $\mathbf{x}$. $I_{\mathbf{x}}$ is a (possibly rotated) rectangular region of the image. This object likelihood is quantified by the certainty of an object detection at $I_{\mathbf{x}}$. The object detector used is a boosted cascade of simple features [Viola01], grouped in N_d classifier stages. At each stage, there is an increasing number of simple features that are selected during training. In the Viola-Jones implementation of the object detector, these are simple Haar-like features. During testing by the classifier at a given stage, different portions of the candidate image patch $I_{\mathbf{x}}$ are tested for matching against these Haar-like features. The qualities of matching between $I_{\mathbf{x}}$ and the features are summed for all features in the stage i, yielding the sum $s_i^{(\mathbf{x})}$. $I_{\mathbf{x}}$ is tested against the classifier at each stage i, if and only if it has been accepted by the previous stage. Acceptance by stage i is granted if the sum $s_i^{(\mathbf{x})}$ exceeds a threshold value τ_i determined during training. The ratio $\left(s_i^{(\mathbf{x})}-\tau_i\right)/|\tau_i|$ quantifies how easily $I_{\mathbf{x}}$ is accepted by the classifier in stage i. In practice, this ratio is always smaller than unity. Then, if $I_{\mathbf{x}}$ is accepted by the classifier in up to stage $N \le N_d$, the certainty of object presence is given by:

$$L_{obj}(\mathbf{x}) = \sum_{i=0}^{N-1}\left(1+10\frac{s_i^{(\mathbf{x})}-\tau_i}{|\tau_i|}\right)10^{i-N_d} \tag{13}$$

It is $L_{obj}(\mathbf{x}) \in [0,1]$ and $L_{obj}(\mathbf{x}) \ge 10^{i-N_d}$ if $I_{\mathbf{x}}$ is accepted by the classifier in stage $i-1$. It is obvious from its definition, that the object presence certainty is largely determined by the ease of acceptance of $I_{\mathbf{x}}$ by the highest stage N it has been accepted. Secure face detections are those accepted by the fi-

nal detector stage N_d. These have a likelihood largely determined by the last term of the sum, $0.1+\left(s_{N_d-1}^{(\mathbf{x})}-\tau_{N_d-1}\right)/\left|\tau_{N_d-1}\right|$.

To define the object likelihood model, firstly the object is sought around $\mathbf{x}$, by perturbing both the space and the size dimensions of the state by an amount of 5% and averaging the resulting object presence certainties L_o. Then, as justified in [Perez04], the averaged object presence certainty is assumed to be exponentially distributed:

$$p\left(\mathbf{y}_{obj}\left|\mathbf{x}\right.\right)\propto\exp\left(-\frac{L_{obj}^{(\max)}-L'_{obj}\left(\mathbf{x}\right)}{2\sigma_{obj}^2}\right) \tag{14}$$

where σ_{obj}^2 is the variance of the distribution, $L_{obj}^{(\max)}$ is an empirically defined maximum object detection certainty (which is classifier-dependent), and $L'_{obj}\left(\mathbf{x}\right)$ is a saturated version of the averaged certainty, such as:

$$L'_{obj}\left(\mathbf{x}\right)=\begin{cases}L_{obj}\left(\mathbf{x}\right) & \text{if } L_{obj}\left(\mathbf{x}\right)\le L_{obj}^{(\max)}\\ L_{obj}^{(\max)} & \text{if } L_{obj}\left(\mathbf{x}\right)>L_{obj}^{(\max)}\end{cases} \tag{15}$$

Small values of the variance σ_{obj}^2 ensure that minor differences in the certainty $L_{obj}\left(\mathbf{x}\right)$ are significantly weighted in the likelihood $p\left(\mathbf{y}_{obj}\left|\mathbf{x}\right.\right)$, thus increasing its selectivity.

When used in the proposed particle filter face tracker, the detected object is a face and the state $\mathbf{x}$ is mapped into the image portion $I_{\mathbf{x}}$ which is the rectangle defined by the state's centre coordinates and size. A typical such region is shown as the red "particle & face detection" rectangle in Figure 14.

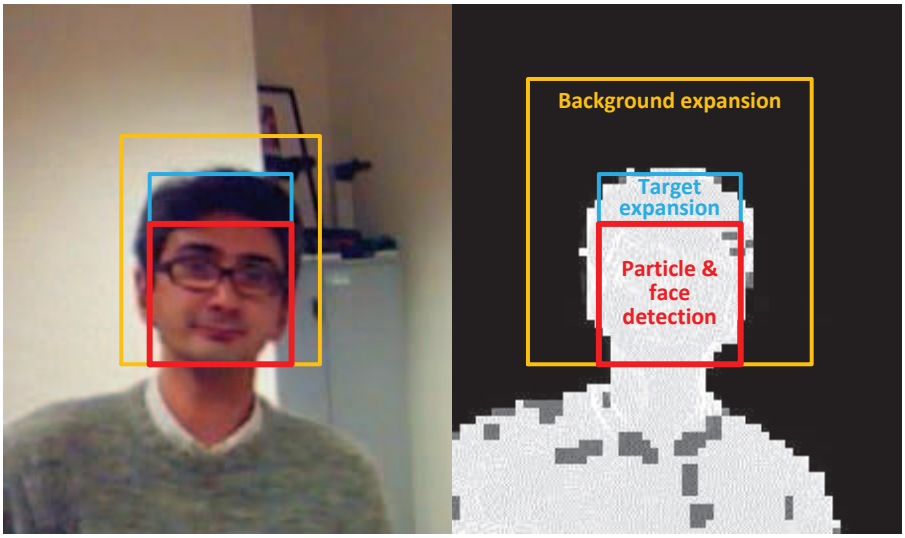

Figure 14: Particle, target expansion and background regions used in the three measurement cues, superimposed on a portion of the face and the foreground evidence image.

The object presence likelihood depends on the model of the object, as this is described by the object detector. This model is trained off-line, and is generic to accurately describe the whole class of similar objects. An example is the face detector, trained to detect all human faces. Note on the other hand that there exist approaches where the detector is updated, being trained by an on-line version of the boosting algorithm [Babenko09]. In this case, the model is object specific, and can serve to discriminate be-

tween two different instances of the object class. For example, two faces present can be discriminated by the tracker if the detectors for each are updated on-line.

4.2.2 Foreground

Foreground evidence is collected from both short-term motion, i.e. the absolute grayscale frame-by-frame difference I_{df}, and medium-term motion, i.e. the binary foreground mask I_{frg} obtained by the adaptive foreground segmentation algorithm discussed in Section 3.2. The foreground evidence image is then given by:

$$I_{fev} = a_{df} I_{df} + a_{PPM} \left(1 - I_{PPM}\right) + a_{frg} I_{frg} \tag{16}$$

where a_{df}, a_{PPM} and a_{frg} are scaling constants. The PPM (introduced in section 3.2.4) is the more robust term in the sum, while the addition of the binary mask amplifies the effect of motion of significant objects in the scene. The frame-by-frame difference term amplifies edges and is important in cases where the foreground estimate is corrupted by a recent lighting change, one that has not yet been able to be learnt into the background.

The mapping of the state $\mathbf{x}$ into an image patch $I_{\mathbf{x}}$ is now done differently. Since the foreground object is the head and not the face, $I_{\mathbf{x}}$ extends beyond the face to the hair region, as shown by the cyan "target expansion" rectangle in Figure 14. The foreground evidence for state $\mathbf{x}$ is gathered from the equivalent portion of the foreground evidence image I_{fev} shown on the right portion of Figure 14. To force the patch to be large enough to include all the moving object, negative foreground evidence is also collected in a region surrounding $I_{\mathbf{x}}$, designated $I'_{\mathbf{x}}$. This is the portion of the surrounding orange "background expansion" rectangle not occupied by the red "particle" and cyan "target expansion" ones in Figure 14. The background expansion is a significant proportion of the target size; suggested values should be around 50% to the left, right and top. The foreground evidence is then defined as:

$$L_{frg}\left(\mathbf{x}\right) = \frac{1}{A\left(I_{\mathbf{x}}\right)} \sum_{i \in I_{\mathbf{x}}} I_{fev}\left(i\right) - \frac{1}{A\left(I'_{\mathbf{x}}\right)} \sum_{i \in I'_{\mathbf{x}}} I_{fev}\left(i\right) \tag{17}$$

where $A(I)$ is the area in pixels of the image patch I, and i is a pixel index for an image patch, denoted one-dimensional for notation simplicity. Note that if the regions $I_{\mathbf{x}}$ and $I'_{\mathbf{x}}$ are non-rotated rectangles, then the foreground evidence is efficiently calculated using the integral image [Viola01] of I_{fev}.

To define the foreground likelihood model, the foreground evidence is assumed to be exponentially distributed:

$$p\left(\mathbf{y}_{frg} \middle| \mathbf{x}\right) \propto \exp\left(-\frac{L_{frg}^{(\max)} - L'_{frg}\left(\mathbf{x}\right)}{2\sigma_{frg}^2}\right) \tag{18}$$

where σ_{frg}^2 is the variance of the distribution, $L_{frg}^{(\max)}$ is an empirically defined maximum foreground evidence, and $L'_{frg}\left(\mathbf{x}\right)$ is a saturated version of the foreground evidence, such as:

$$L'_{frg}\left(\mathbf{x}\right) = \begin{cases} L_{frg}\left(\mathbf{x}\right) & \text{if } L_{frg}\left(\mathbf{x}\right) \le L_{frg}^{(\max)} \\ L_{frg}^{(\max)} & \text{if } L_{frg}\left(\mathbf{x}\right) > L_{frg}^{(\max)} \end{cases} \tag{19}$$

The foreground likelihood does not depend on any model that is specific to the foreground object being tracked. It only employs a general background model that depends on the variations of the pixels in the video frames. On the other hand, the background modelling process is not entirely agnostic to objects,

since the learning rate of the pixels corresponding to objects is lowered accordingly to decelerate fading of foreground objects to the background.

4.2.3 Colour matching

Colour likelihood represents the degree of similarity between a colour model of the image patch $I_{\mathbf{x}}$ corresponding to the state $\mathbf{x}$ and the colour model of the target. As in the case of foreground measurement, the image patch is expanded at the top to include the hair region, as shown on the left portion of Figure 14. Both models are represented as one-dimensional colour histograms. The similarity is enumerated by the Bhattacharyya coefficient between the two models.

To calculate the histograms, the red (R), green (G) and blue (B) colour components are first quantised to N_h levels and then are combined into a one-dimensional colour quantity $c(R,G,B)$, defined as:

$$c(R,G,B) \equiv \left\lfloor R\frac{N_h}{256} \right\rfloor + N_h \left\lfloor G\frac{N_h}{256} \right\rfloor + N_h^2 \left\lfloor B\frac{N_h}{256} \right\rfloor \tag{20}$$

where $\lfloor a \rfloor$ denotes the largest integer smaller than or equal to a . The histogram of $c(R,G,B)$ is then calculated. Since $\{R,G,B\} \in [0,255]$, the one-dimensional colour quantity $c(R,G,B)$ is an integer in the range $[0,\ldots,N_h^3-1]$, yielding N_h^3 histogram bins.

The candidate image patch colour histogram $\mathbf{h}_{\mathbf{x}}$ is constructed from the pixels in $c(R,G,B)$ corresponding to $I_{\mathbf{x}}$. The colours found there should match the target model, but any others found in the immediate surroundings should not, if the target is completely included in the patch. To measure this, a "background expansion" rectangle is again introduced, similar to the one used in foreground measurement, but with a smaller expansion factor of around 20% (contrast the orange "background expansion" rectangles on the left and the right parts of Figure 14. Another colour histogram $\mathbf{h}'_{\mathbf{x}}$ is constructed from the pixels corresponding to a region surrounding $I_{\mathbf{x}}$, designated $I'_{\mathbf{x}}$.

Comparing histograms $\mathbf{h}_{\mathbf{x}}$ and $\mathbf{h}'_{\mathbf{x}}$ to the target model one $\mathbf{h}_{ref}$ is done using the respective Bhattacharyya coefficients. If all of the tracked object is included in $I_{\mathbf{x}}$, without any background, then ideally $\mathbf{h}_{\mathbf{x}}$ is similar to $\mathbf{h}_{ref}$, while the surrounding colours are different and $\mathbf{h}'_{\mathbf{x}}$ is not similar to $\mathbf{h}_{ref}$. Thus the colour similarity metric is defined as:

$$L_c\left(\mathbf{x}\middle|\mathbf{h}_{ref}\right) = \sum_{i=0}^{N_h^3-1} \sqrt{\mathbf{h}_{\mathbf{x}}(i)\mathbf{h}_{ref}(i)} - \sum_{i=0}^{N_h^3-1} \sqrt{\mathbf{h}'_{\mathbf{x}}(i)\mathbf{h}_{ref}(i)} \tag{21}$$

where $\mathbf{h}_{\mathbf{x}}(i)$, $\mathbf{h}'_{\mathbf{x}}(i)$ and $\mathbf{h}_{ref}(i)$ are the i -th bins of the candidate image patch, its surroundings and the reference histograms respectively.

To define the colour likelihood model, the colour similarity metric $L_c\left(\mathbf{x}\middle|\mathbf{h}_{ref}\right)$ is assumed to be exponentially distributed:

$$p\left(\mathbf{y}_c\middle|\mathbf{x}\right) \propto \exp\left(-\frac{L_c^{(\max)} - L'_c\left(\mathbf{x}\middle|\mathbf{h}_{ref}\right)}{2\sigma_c^2}\right) \tag{22}$$

where σ_c^2 is the variance of the distribution, $L_c^{(\max)}$ is an empirically defined maximum colour similarity, and $L_c'\left(\mathbf{x}\middle|\mathbf{h}_{ref}\right)$ is a saturated version of the colour similarity metric, such as:

$$L_c'\left(\mathbf{x}\middle|\mathbf{h}_{ref}\right)=\begin{cases}L_c\left(\mathbf{x}\middle|\mathbf{h}_{ref}\right) & \text{if } L_c\left(\mathbf{x}\middle|\mathbf{h}_{ref}\right)\le L_c^{(\max)}\\ L_c^{(\max)} & \text{if } L_c\left(\mathbf{x}\middle|\mathbf{h}_{ref}\right)> L_c^{(\max)}\end{cases} \tag{23}$$

Since colour similarity is measured using Bhattacharyya coefficients that have a maximum value of unity, we select $L_c^{(\max)}=1$.

The colour likelihood depends on the colour model of the target, $\mathbf{h}_{ref}$, as this is trained at initialization of the target, and possibly updated during the course of the track. In order to increase the discriminating ability of the target model, the reference histogram is trained by attenuating the effect of the colours in the immediate background of the target. To do so the "background expansion" rectangle is again used. Given the region I_{init} where the target is initialised and its expansion to the surrounding area, I_{init}', three histograms are calculated: $\mathbf{h}_{init}$ from the frame at region I_{init}, $\mathbf{h}_{ext}$ from the frame at region I_{init}' and $\mathbf{h}_{bkg}$ from the background estimation at region I_{init}. The i-th bin of the immediate background and surrounding colour histogram is then calculated as:

$$\mathbf{h}_{bkg}'\left(i\right)=\frac{\mathbf{h}_{bkg}\left(i\right)\mathbf{h}_{ext}\left(i\right)}{\sum_{k=0}^{N_b^3-1}\mathbf{h}_{bkg}\left(k\right)\mathbf{h}_{ext}\left(k\right)} \tag{24}$$

To attenuate colours that are present in $\mathbf{h}_{bkg}'$ from $\mathbf{h}_{init}$, the histogram bins of the target model $\mathbf{h}_{ref}\left(i\right)$ are defined as $\mathbf{h}_{ref}\left(i\right)=f\left(i\right)\cdot\mathbf{h}_{init}\left(i\right)$, i.e. the bin values $\mathbf{h}_{init}\left(i\right)$ are modified by a multiplicative factor $f\left(i\right)$ defined as:

$$f\left(i\right)=\begin{cases}\min\left(\mathbf{h}_{bkg}'\right)/\mathbf{h}_{bkg}'\left(i\right) & \text{if } \mathbf{h}_{bkg}'\left(i\right)>\mathbf{h}_{init}\left(i\right)/K\\ 1 & \text{elsewhere}\end{cases} \tag{25}$$

This factor is smaller than unity, effectively attenuating the respective histogram bin values when the respective colour is more evident in the background than in the initialization region. The count of the background histogram bin $\mathbf{h}_{bkg}'\left(i\right)$ relative to the target one $\mathbf{h}_{init}\left(i\right)$ to be able to affect $f\left(i\right)$ is set by the constant $K>1$.

4.3 Particle filter tracker

4.3.1 Particle filter implementation

The SMART face tracker is built utilising particle filters. The weight for particle $\mathbf{x}$ is updated based on the measured likelihood $p\left(\mathbf{y}\middle|\mathbf{x}\right)$, where $\mathbf{y}$ is any of the object detection, foreground and colour measurements introduced in the previous section. The filter is a Sequential Importance Resampling one, where systematic resampling [Kitagawa96] is employed. As a SIR particle filter, the selected proposal distribution is the same as the object model. We try three different object models, as discussed in section 4.3.2.

The face detection, foreground and colour matching likelihoods introduced in section 4.2 are combined

together into a fused likelihood by multiplying together their clipped versions. Clipping is done at the smaller values of the likelihoods, to avoid the very small value of some likelihood voiding the normal values of others. Clipping is done at a fraction k_c of the largest value of the particle likelihoods.

Finally, multiple target management is discussed in section 4.3.3.

4.3.2 Object model

Humans, especially when moving indoors, tend to do so unpredictably. Hence a suitable object model for tracking people should make only a weak assumption for state evolution. Motion smoothness is guaranteed by a Gaussian random walk component centred at the previous state $\mathbf{x}_{n-1}$, with covariance $\mathbf{C}_\mathbf{x}$:

$$p\left(\mathbf{x}_n \left| \mathbf{x}_{n-1}\right.\right) = N\left(\mathbf{x}_n \left| \mathbf{x}_{n-1}, \mathbf{C}_\mathbf{x}\right.\right) \tag{26}$$

The covariance matric $\mathbf{C}_\mathbf{x}$ is not constant. It depends on the estimated target size at the previous time step. It is diagonal, with position and size terms being fractions of the estimated target size. The position terms have larger fractions of the target size, since targets closer to the camera are expected to move faster in terms of frame pixels.

The object model should also provide for lock recovery after target loss (due to erratic motion or occlusion). This is aided by a uniform component, as in [Perez04]:

$$p\left(\mathbf{x}_n \left| \mathbf{x}_{n-1}\right.\right) = \beta_{RW} N\left(\mathbf{x}_n \left| \mathbf{x}_{n-1}, \mathbf{C}_\mathbf{x}\right.\right) + \left(1-\beta_{RW}\right) U\left(\mathbf{x}_n \left| V\right.\right) \tag{27}$$

where $U\left(\mathbf{x}\left|V\right.\right)$ is the uniform distribution in some volume V of the state-space, and β_{RW} is the relative weight of the random walk (multivariate Gaussian) component. β_{RW} is usually close to unity, allowing only a few of the particles to abandon the smooth motion of the random walk component and embark into spanning the volume V, trying to reacquire targets that might be lost. The volume V spans four times the standard deviation of the face size in $\mathbf{C}_\mathbf{x}$.

Blindly aiming at lock recovery in cases of target loss with a uniform component in the proposal distribution and the object model has two drawbacks:

- Subspaces of the state-space are randomly (uniformly) searched for evidence of the target. Most of these uniformly distributed particles are finding nothing.
- Particles are dedicated to the uniform component even when there is no target lock loss.

It is therefore dangerous to allocate too many of the particles to the uniform component, hence its relative weight $\left(1-\beta_{RW}\right)$ in (27) needs to be small and lock recovery is not guaranteed.

Instead of utilising the Gaussian plus uniform mixture, we build a measurement-assisted object model by following the approach of [Perez04] for optimum proposal distribution design: we employ the Gaussian random walk component, plus the contribution of the measurement model in the form of a sum of Gaussian densities with relative weight $(1-\beta_{RW})$. To establish the contribution of the measurement model, a grid $\mathbf{x}_g$ around all the particles is searched for the n_g locations $\left\{\mathbf{x}_g^{(k)}\right\}_{k=1}^{n_g}$ of good measurement match.

At this point we deviate from the approach of [Perez04] in two ways:

- We use this measurement-assisted proposal for the object model as well, hence the resulting particle filter tracker remains an SIR one.
- We do not use a constant threshold to define good measurement match locations $\mathbf{x}_g^{(k)}$. Instead, we use a fraction τ_g of the best match obtained at the previous time instance, i.e. the maxi-

mum of the particle likelihoods $p\left(\mathbf{y}_{n-1}\left|\mathbf{x}_{n-1}^{(i)}\right.\right)$:

$$p\left(\mathbf{y}_n\left|\mathbf{x}_g^{(k)}\right.\right) > \tau_g * p\left(\mathbf{y}_{n-1}\left|\mathbf{x}_{n-1}^{(i)}\right.\right) \tag{28}$$

- We do not just use colour as the measurement cue employed for modifying the measurement-assisted proposal distribution. Instead we use all the combinations of face presence, foreground and colour matching measurements employed in our multi-cue visual face trackers to find the good matching locations on the grid.

The locations $\mathbf{x}_g^{(k)}$ of good measurement match are used to bias the proposal distribution towards them. This is achieved by forming a sum of n_g Gaussians centred at the good grid locations and with covariance matrices $\mathbf{C}_\mathbf{x}$. Hence the object model and the proposal distribution are given by:

$$p\left(\mathbf{x}_n\left|\mathbf{x}_{n-1}\right.\right) = q\left(\mathbf{x}_n\left|\mathbf{x}_{n-1},\mathbf{y}_n\right.\right) = \beta_{RW} N\left(\mathbf{x}_n\left|\mathbf{x}_{n-1},\mathbf{C}_\mathbf{x}\right.\right) + \frac{1-\beta_{RW}}{n_g}\sum_{k=1}^{n_g} N\left(\mathbf{x}_n\left|\mathbf{x}_g^{(k)},\mathbf{C}_\mathbf{x}\right.\right) \tag{29}$$

The difference is performance between the three choices for object model and proposal distribution expressed by equations (26), (27) and (29) is assessed in section 4.4.

4.3.3 Target management

From an algorithmic point of view, a particle filter tracker updates the previous state to the current one, represented by a set of particles. From a system point of view, the tracker needs to initialise, maintain and eventually terminate multiple targets. Multiple visual targets in the SMART face tracking system are handled by assigning an independent Particle Filter (PF) tracker to each of them.

In this section we describe a loop of the tracking system, i.e. the operations happening with every new frame being processed. The tracking algorithm discussed thus far is only a part of the complete system.

- Coarse face detection: This is carried out only at the foreground parts of the frame, using a restricted set of locations and scales, to ensure fast operation. The set of locations is determined from a training sequence and the scale depends on the vertical displacement in the frame. This displacement is due to the distance from the camera (which we want to model), but also due to the height of the person (which is the unwanted variation, since we want our model to capture everybody). The modelling is done by collecting all detected faces from the training sequence and finding their size as a linear function of their vertical displacement. At every frame, initialisation is attempted at some vertical offsets, with nominal size the one given by the model, and some more up- and down-scaled sizes around the nominal size. The detected faces are termed contacts and are used to initialise or update targets.
- Target-contact association: Subsequently the contacts are associated with existing targets based on the Euclidean distance of their coordinates, normalised by the width of the target. Only pairs not exceeding a maximum association distance are associated. An optimal greedy algorithm, the Hungarian (or Munkres) algorithm [Blackman99] is used for the association. The algorithm minimises the overall distance between the targets and the contacts. In our code we use the excellent implementation of the algorithm for C found in [Buehren09].
- Target initialisation: Un-associated contacts are used for target initialisation. This involves the initialisation of a new particle filter tracker and the training of the target colour model.
- Target update using contact: Associated contacts are used to update the target. In our current SMART system the update involves a retraining of the colour model using some memory. One can also consider resetting the target state based on the contact.
- Target tracking: All targets except those just initialised are tracked using the particle filter tracker presented thus far. Tracking can also involve updating the target models for some of the cues based on the measurement likelihoods of the other cues. Should for instance the object detection cue yield a very high likelihood (indicative of the correct tracking of a frontal face), then the colour model can be updated. We do not use this mechanism in the current SMART

face tracker, since this update is done using the associated contacts.

- Target hiatus and termination: During tracking the measurement likelihoods for all cues are monitored, and the number of consecutive frames where they are too low is counted. The low likelihood thresholds can vary linearly between a strict and a loose one, as a function of the age of the target. In this way younger targets are handled differently to older ones. When the target models do not match well with the evidence collected it is an indication of change in appearance, tracker failure or disappearance of the target from the scene. Targets with momentarily low matching qualities enter a hiatus state where they are updated but are not reported, while those for which the low quality persists are terminated. Target termination is also based on particle spread. A large spread indicates particles with no reason to lock in a specific frame location, hence they are spread apart by means of the random components of the object model. Too large spread indicates lack of target lock in a different way than the measurement likelihood. In cases of measurement ambiguity (e.g. background colours similar to those of the target), the colour likelihood can be high in a whole frame region but at the same time the particles exhibit high spread.

4.4 Results

Seven particle filter face trackers are built utilising all possible single, pair and triple combinations of the likelihood functions introduced. We also have built a deterministic tracker based on CAM-Shift and a baseline colour particle filter tracker.

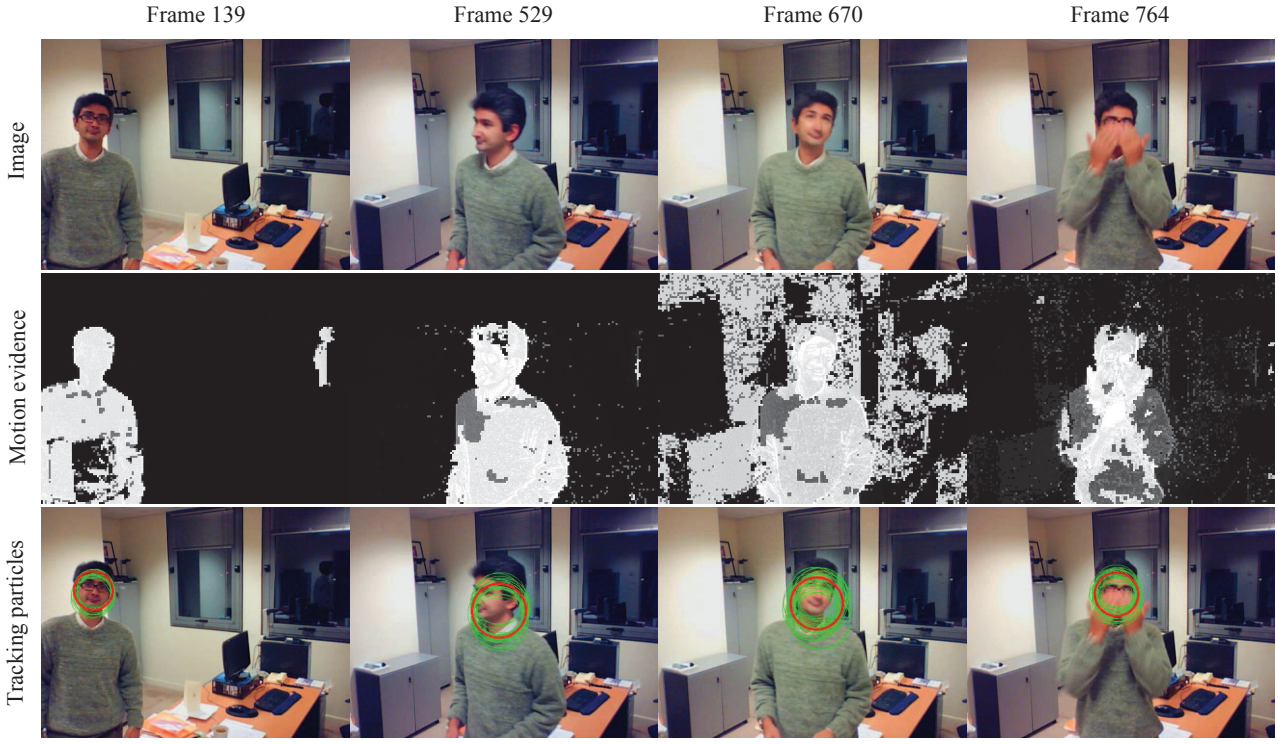

Figure 15: Example frames in test sequence, the resulting foreground evidence and the operation of the tracker. The frames depict normal conditions, profile face, camera exposure change and occlusion.

The trackers are tested on an annotated sequence that exhibits face in-plane rotations, out-of-plane pose variations, illumination variations and occlusions, as well as global intensity variations due to changes in the exposure of the camera. The face is visible (albeit sometimes occluded) in 3,000 frames (120 sec at 25 frames per sec). The person moves a lot, hence the standard deviations of the horizontal displacement, vertical displacement and size are 191%, 30% and 36% of the average face size respectively. Examples of the conditions encountered in the sequence are shown in the top row of Figure 15.

In frame 139 the conditions are ideal: the colour model is only recently initialised, the foreground measurement is noise-free and the pose of the face is easy to detect. In frame 529 the colours have not changed, the foreground measurement has some noise (stray pixels) but more significantly the pose is profile, making face detection quite difficult. In Frame 670 the colours have changed significantly due to the change of the camera exposure. For the same reason the foreground measurement is showing many false blobs that need to be learnt back into the background. The face is somewhat diagonally tilted, but not enough to cause detection problems. Finally, in Frame 764 there is severe occlusion rendering detection very difficult, while there is still quite some foreground measurement noise, since only 100 frames have passed since the exposure change. The occluding hands and the visible upper part of the face ensure that colour measurement gives a good match.

4.4.1 Likelihood sensitivity

We begin the evaluation of the tracker by considering how sensitive the likelihoods introduced in section 4.2 are to offsets around the ideal state. We consider the different situations depicted in Figure 15. For this evaluation an offset is introduced in the horizontal direction from the actual face position. The likelihoods should drop away from the actual position (offset 0), but should do so rather progressively, to be able to attract the particles when they are a bit close. The three single-cue likelihood variations are depicted in Figure 16 for colour, Figure 17 for foreground and Figure 18 for face. The foreground likelihood is the smoothest of them all, while the face detection one is quite irregular. All of them have the profile peak at an offset from the correct position. Finally, the fused triple-cue likelihood is then in Figure 19. The fused likelihood has a selectivity that is several orders of magnitude larger that the single-cue ones. In all the single-cue (and the fused) likelihoods in the profile case the peak is off-centre, which is expected, as the tracker tends to track the head and not the profile face in such cases.

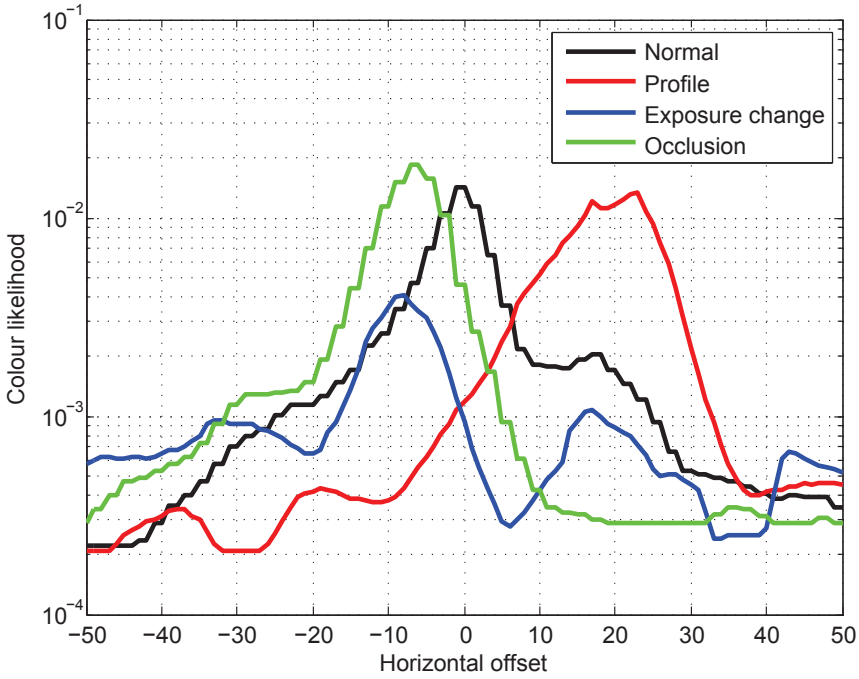

Figure 16: Colour matching likelihood for the four conditions in the example frames.

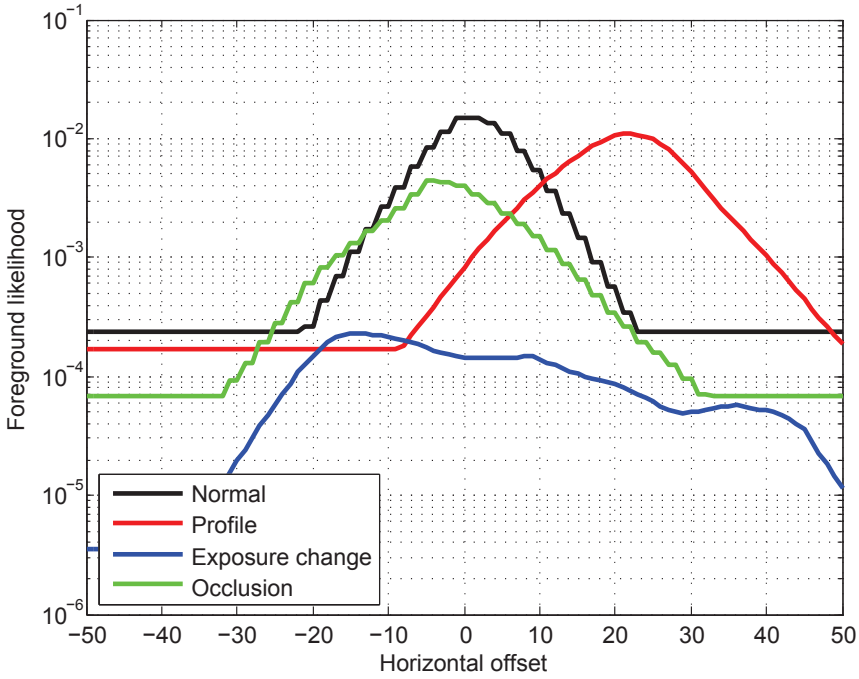

Figure 17: Foreground likelihood for the four conditions in the example frames.

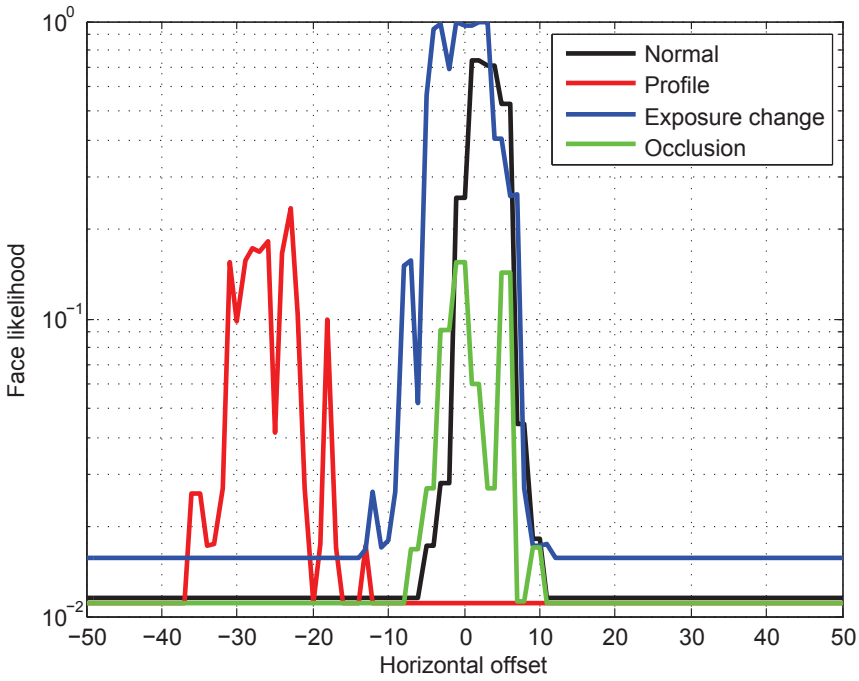

Figure 18: Face detection likelihood for the four conditions in the example frames.

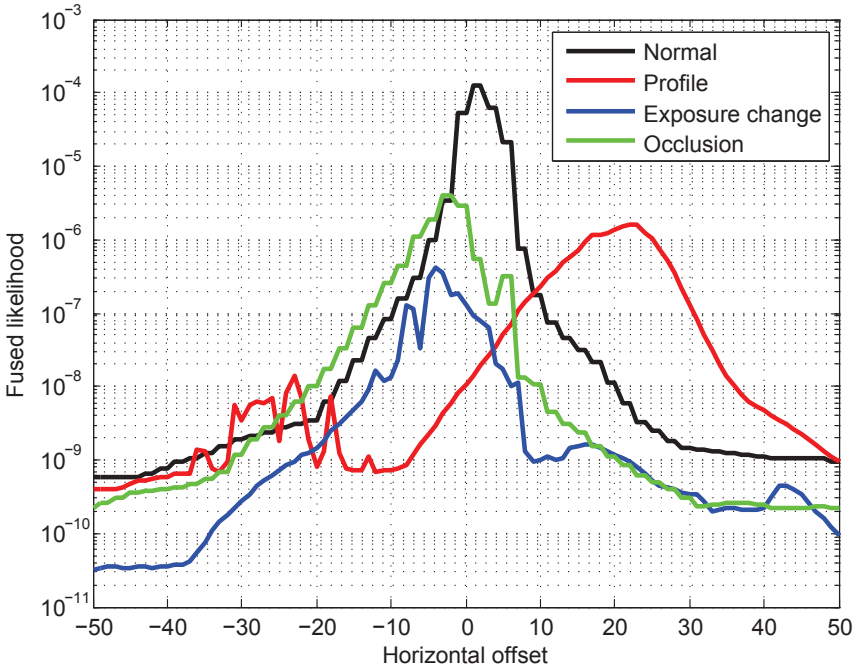

Figure 19: Fused likelihood for the four conditions in the example frames.

4.4.2 Evaluation metrics and comparison to baseline

We define the relative tracking error as the sum of the square of the horizontal, vertical and size errors relative to the face width. We also consider the current face as being tracked if the relative tracking error is smaller than unity. The hit rate of the tracker is the percentage of frames the face is being tracked. The tracking quality is determined by the root-mean of the relative tracking error, averaged over the frames tracking is considered successful.

Given the metrics introduced above, we can quantify the performance of the trackers. Table 2 summarises the hit rate and the relative tracking error achieved by the seven trackers. The latter is given both for the frames the tracker operates well and for all the frames.

Table 2: Effect of the measurement combination to the particle filter tracker. The measurement-assisted object model and 150% background expansion are used.

Measurements			Hit rate (%)	RMS error (%)	
Object presence	Foreground	Colour matching		Total	Only at hits
Yes	No	No	64.67	198.47	31.87
No	Yes	No	81.85	93.89	24.21
No	No	Yes	100.0	26.24	
Yes	Yes	No	88.77	81.47	22.49
No	Yes	Yes	100.0	22.12	
Yes	No	Yes	100.0	15.92	
Yes	Yes	Yes	100.0	15.39	

Face detection as tracking measurement performs worst since it is the least persistent of all cues; any pose variation causes failure. Foreground measurements follow. Even though this measurement is quite persistent, lighting changes corrupt the measurement and there is no discriminating ability be-

tween the face, other parts of the body and moving background. Hence any occlusion of the face can force the particles to the body, where they do not spread as there is significant foreground evidence, so they cannot easily recover back to the face. Colour performs best since is by far the most discriminant cue, even though it lacks persistence when lighting changes occur.

We then compare in Table 3 the two baseline trackers to the tracking performance we achieved with our first face tracker on January 2012 and the latest one of February 2013.

Table 3: Comparison of the baseline, the early SMART and the mature SMART face trackers.

Tracker	Hit rate (%)	Total RMS error (%)
Baseline: CAM-Shift (Foreground & background colour modelling in Cb,Cr with 16 levels per colour component)	98.58	31.15
Baseline: Colour PF (Foreground only colour modelling in Y,Cb,Cr with 16 levels per colour component, Gaussian proposal)	94.19	49.86
January 2012 triple-cue PF (Foreground & background colour modelling in Y,Cb,Cr with 8 levels per component, frame difference only, Gaussian & Uniform proposal)	95.37	30.89
February 2013 colour PF (Foreground & background colour modelling in Y,Cb,Cr with 16 levels per component, Gaussian & measurement-assisted proposal)	100.0	26.24
February 2013 triple-cue PF (Foreground & background colour modelling in Y,Cb,Cr with 16 levels per component, full foreground modelling, Gaussian & measurement-assisted proposal)	100.0	15.39

Obviously our face trackers exhibit large performance boost compared to the colour PF baseline one and significant boost over the CAM-Shift one. Note that the fact CAM-Shift surpasses the early PF trackers is artificial. The CAM-Shift tracker was the latest implemented (April 2013), and there we tried intensity-invariant colour matching (after the suggestion of DSP2013 reviewers for [Pnevmatikakis13]) and a better target re-initialisation scheme. Due to lack of time, these changes have not been included in the PF trackers, a task left for the M30 systems to be reported in the second version of this document.

The main differences between our early prototype and the current near-final version are:

- We underestimated the effect of number of levels per colour component. 8 are too few for a good discriminating ability. We do get the latter by using 16.
- We underestimated the effect of using larger background expansion factors.
- Pure motion (as modelled by frame-by-frame difference) is not as persistent as the combination of motion with foreground modelling.
- Proper proposal distribution and object model selection plays a significant role in performance.

In the following paragraphs we consider the effect of these changes and all other tracker parameters in detail, using the metrics introduced here.

4.4.3 Likelihood selectivity: Measurement model variances

We begin by analysing the effect of the measurement model variances. Smaller values penalise discrepancies from the respective models, making the likelihood functions very selective. Larger values at-

tenuate model differences, making the likelihood functions more forgiving to differences. A balance needs to be achieved: Selectivity is wanted, or else the particles will spread everywhere in the state-space, but too much selectivity can give negligible likelihood functions when the target starts appearing a bit different than the model. Certainly higher selectivity can be allowed to measurements that are very discriminative.

The colour likelihood selectivity affects colour tracker performance as shown in Table 4 and in Figure 20. The wanted selectivity balance is easily achieved, as there is an optimum performance range in [0.03,0.1].

Table 4: Effect of colour likelihood selectivity, as expressed by colour model variance, on single-cue colour tracker performance.

Colour model variance	Hit rate (%)	RMS error (%)	
		Total	Only at hits
2^{-1}	19.29	859.9	52.33
2^{-2}	100.0	32.68	
2^{-3}	100.0	26.72	
2^{-4}	100.0	26.24	
2^{-5}	100.0	26.91	
2^{-6}	100.0	27.55	
2^{-7}	93.58	54.08	33.81

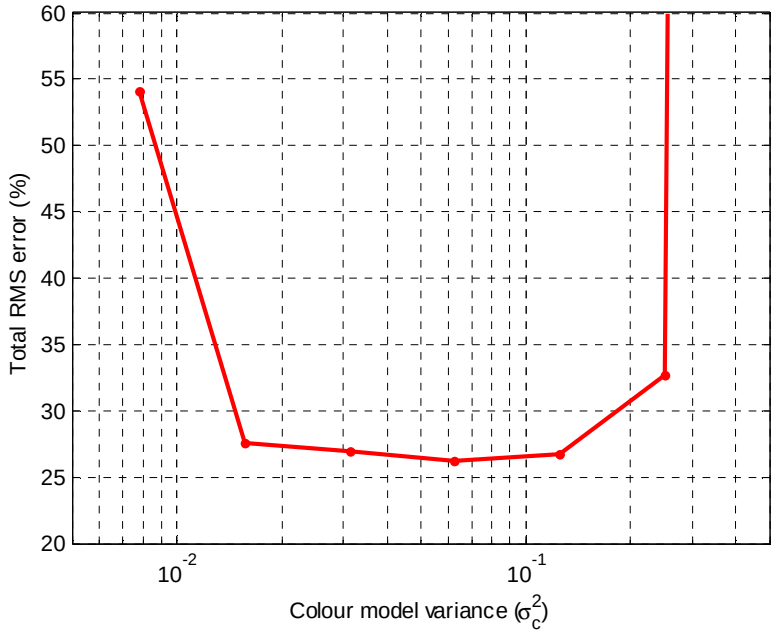

Figure 20: Effect of colour likelihood selectivity on single-cue colour tracker performance.

The situation for the foreground measurement is not as straight-forward, since the tracker locks to human bodies when the faces are occluded, and the performance remains bad, fluctuating meaninglessly, no matter our choice of foreground measurement selectivity. Instead we assess the effect of foreground measurement selectivity to the dual-cue colour plus foreground tracker. This tracker yields correct results and allows the evaluation of the effect of foreground measurement selectivity. The results are shown in Figure 21.

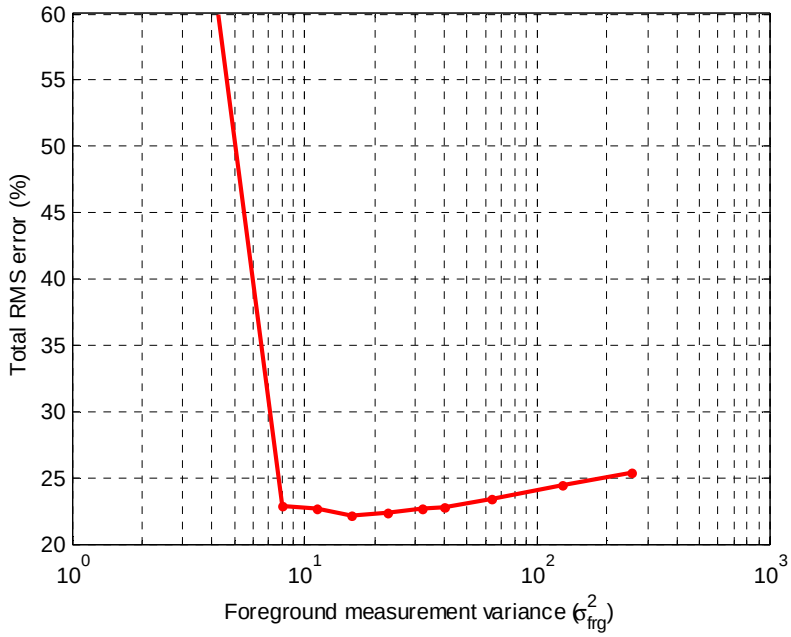

Figure 21: Effect of foreground likelihood selectivity on dual-cue colour plus foreground tracker performance.

Finally, the effect of object detection measurement selectivity on the dual-cue colour plus object presence tracker is shown in Figure 22.

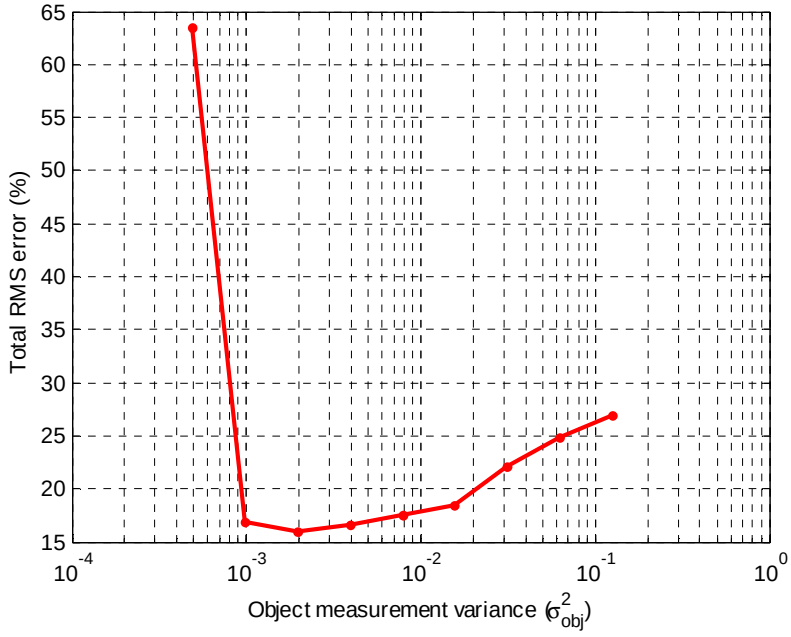

Figure 22: Effect of object presence likelihood selectivity on dual-cue colour plus object presence tracker performance.

The excellent behaviour at very high object detection selectivity values is justified, since object presence is the most discriminating measurement of the three, albeit the least persistent. We want the particles to tightly fit around the state vector that yields high object presence likelihood, since we are certain

that there the tracker operates correctly. When the pose, illumination or expression render object presence measurement likelihood small, then we wish the particles to spread out, in turn being constrained by the other, less discriminative but more persistent measurements. The only exception to this rule is when two objects of the same class are close to each other (two faces) or an image patch resembling a face is close to the target. Then, should the target no longer resemble the object, the tracker will be eager to select the false nearby object. This happens with very large selectivity in our tracker, and causes the increase of RMS error for very small values of the object presence variance.

4.4.4 Effect of system parameters

Having established the effect of likelihood selectivity, we now turn our attention to the various system parameters of the face tracker. First we examine the effect of number of histogram levels per colour component in the histograms. For this examination we use the colour PF tracker. The results are shown in Table 5. As expected, 8 histogram levels per colour component are grossly insufficient. The tracker barely becomes stable at 10, while optimum performance seems to be obtained between 16 and 24. Larger numbers of levels over train the histogram to the initialisation appearance, reducing discriminating ability under illumination variations. Since increasing the levels increases both processing and memory requirements, we select to use 16 of them.

Table 5: Effect of the number of levels per colour component in the histograms on the colour matching particle filter tracker.

Levels	Hit rate (%)	Total RMS error (%)
8	69.45	208.6
10	99.57	31.58
12	99.89	28.53
16	100.0	26.91
24	100.0	26.56
32	100.0	27.38

Then we examine the effect of the object model and proposal distribution design. For this examination we again use the colour PF tracker. The results are shown in Table 6. The success of the proposed measurement-assisted scheme is obvious.

Table 6: Effect of the object model on a colour matching particle filter tracker.

Gaussian	Uniform	Measurement	Hit rate (%)	Total RMS error (%)
Yes	No	No	100.0	35.18
Yes	Yes	No	100.0	32.49
Yes	No	Yes	100.0	29.04

Next we examine the effect of the background colour modelling expansion factor. For this examination we again use the colour PF tracker. The results are shown in Figure 23. An expansion between 80% and 170% is close to optimum.

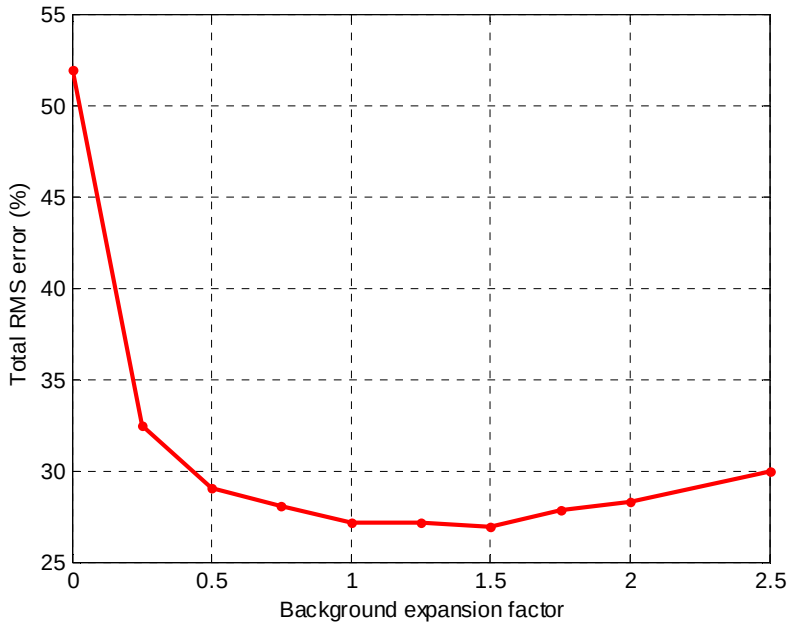

Figure 23: Effect of the background expansion factor on a colour matching particle filter tracker featuring a measurement-assisted object model.

The number of particles is the last parameter whose effect is investigated using the colour PF tracker. The results are shown in Table 7. It appears that even very low number of particles still result to a stable tracker. This is good, since the processing requirements of a PF tracker are directly proportional to the number of involved particles. On the other hand, we are reluctant to reduce the number of particles to extremes, before we further test the tracker in different situations. For the time being we fix the number of particles to 50.

Table 7: Effect of the number of particles on the colour matching particle filter tracker.

Particles	Hit rate (%)	Total RMS error (%)
6	100.0	28.80
12	100.0	27.12
25	100.0	26.87
50	100.0	26.91
100	100.0	26.67

Finally, the effect of motion modelling just by frame difference or also including adaptive foreground is considered. For this, in Table 8 we compare the motion only, face & motion, colour & motion and the face, colour & motion trackers with and without full adaptive foreground modelling. The single cue tracker motion tracker is quite unstable, either with frame difference, of with the complete foreground modelling. The face and motion dual cue tracker is only stable with foreground modelling, while the introduction of colour, either in the dual or triple cue tracker renders it stable in both cases. Then the use of foreground modelling leads to improved accuracy results. Apart from the tracking performance, foreground analysis helps the tracking system in target initialisation, since the face detectors only need to run at the areas where there is suitable foreground evidence. Hence we will be employing the full foreground modelling in our face tracker.

Table 8: Effect of the foreground modelling in trackers involving motion and other cues.

Motion measurement	Frame difference		Frame difference & foreground	
Tracker	Hit rate (%)	Total RMS error (%)	Hit rate (%)	Total RMS error (%)
Motion	91.48	62.50	81.85	93.89
Face & motion	88,88	81.48	81.47	22.49
Colour & motion	100.0	21.97	100.0	21.96
Face, colour & motion	100,0	19.24	100.0	18.85

The only situation that colour is performing worse than the rest cues is the occlusion by similar coloured object, like the hands shown in Figure 24. There the tracking error for the three cues and their fusion are compared, and the frames the moment the face is again fully visible are given. The fusion and the face detection recover from the occlusion instantly. Motion follows but colour is much delayed. In fact these are the only sources of target miss in the colour cue.

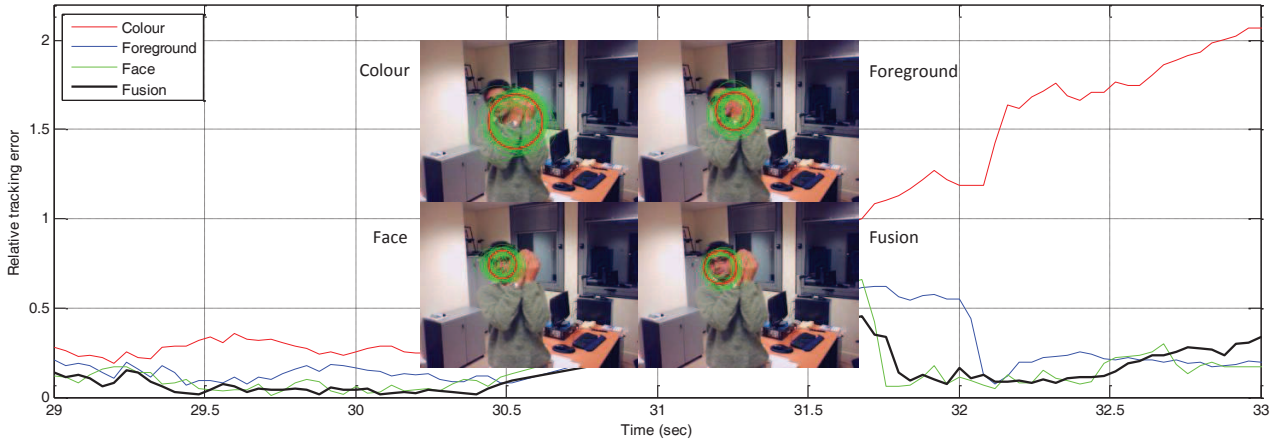

Figure 24: Tracking error and example frames for the different cues during an occlusion by hands.

The dual-cue trackers involving colour reach 100% hit rate, with the combination with foreground performing best in terms of accuracy. Finally, combining all three cues leads again to 100% hit rate, only with even better accuracy.

4.5 Open-source face tracking system

The system described above remains proprietary to the SMART consortium. To facilitate video processing at indoors edge nodes the community sets up, we plan to distribute some open-source face trackers. These will be found under the SampleClients\C_PP directory of our open source release and will be documented in our Trac at:

http://opensoftware.smartfp7.eu/projects/smart/wiki/PerceptualComponents

4.5.1 Current status

Currently we have built a CAM-Shift face tracker that utilises the OpenCV implementation of the algorithm [OpenCVtracking].

4.5.2 Future plans

Our immediate plans foresee the following steps:

- Include target management and release the first open source version.
- Prepare a Kalman version.

4.5.3 Limitations

We do not plan to include any of our proprietary measurement cues, nor our particle filter implementations in the open source face trackers.

5 Conclusions

This document details the two visual processing systems used within SMART. The near-field system performs face tracking, intended to be used indoors or in restricted environments outdoors. The far-filed system performs crowd analysis, by considering crowds as moving foreground pixel blobs. Both these systems remain proprietary to the SMART consortium, but simpler, open source versions have already been made available. Members of the SMART community that are knowledgeable in visual signal processing can use the description of the full versions of the algorithms together with the software implementation of the simpler versions as a starting point to developing their own crowd analysis and face tracking systems.

5.1 Plans for M30

The main drawback of this version of the visual processing algorithms' description is the limited testing sequences. Due to factors beyond our control, the development of this version of the algorithms did not have a hand the actual SMART visual data. This means that our results in sections 3.4 and 4.4 have been obtained after limited testing with the sequences described there. This will change in the second version of the document, as the SMART data will be available sometime in the summer of 2013.

Crowd analysis efforts will focus on speeding up the current implementation, adding a more robust shadow detector and performing more types of motion analysis (converging and diverging).

Face tracking efforts will focus on fine-tuning the target termination criteria and on different strategies for fusing the different measurement types.

6 BIBLIOGRAPHY AND REFERENCES

[Stauffer00] C. Stauffer and W. E. L. Grimson, “Learning patterns of activity using real-time tracking,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 8, pp. 747–757, 2000.

[Pnevmatikakis06] A. Pnevmatikakis and L. Polymenakos, “Robust estimation of background for fixed cameras,” in 15th International Conference on Computing (CIC ’06), Mexico City, Mexico, November 2006, pp. 37–42.

[Xu05] L. Xu, J. Landabaso, and M. Pardas, “Shadow removal with blob-based morphological reconstruction for error correction,” in IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP 2005), Philadelphia, PA, USA, March 2005.

[F1score13] “F1 score”, Wikipedia, http://en.wikipedia.org/wiki/F1_score, May 2013.

[OpenCVMOG2] Gaussian Mixture-based Background - Foreground Segmentation Algorithm, http://docs.opencv.org/trunk/modules/video/doc/motion_analysis_and_object_tracking.html#backgroundsubtractormog2

[Bernardin09] K. Bernardin, R. Stiefelhagen, A. Pnevmatikakis, O. Lanz, A. Brutti, J. R. Casas, and G. Potamianos, “Person tracking,” in Computers in the Human Interaction Loop, ser. Human-Computer Interaction, A. Waibel and R. Stiefelhagen, Eds. Springer, 2009, pp. 11–22.

[Viola01] P. A. Viola and M. J. Jones, “Rapid object detection using a boosted cascade of simple features,” in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), Kauai, HI, USA, December 2001, pp. 511–518.

[Ding09] X. Ding, H. Xu, P. Cui, L. Sun, and S. Yang, “A cascade SVM approach for head-shoulder detection using histograms of oriented gradients,” in IEEE International Symposium on Circuits and Systems (ISCAS 2009), Taipei, Taiwan, May 2009, pp. 1791–1794.

[Waibel04] A. Waibel, H. Steusloff, and R. Stiefelhagen, “CHIL: Computers in the human interaction loop,” in 5th International Workshop on Image Analysis for Multimedia Interactive Services, Lisboa, Portugal, April 2004, pp. 175–178.

[CHIL] “CHIL (computer in the human interaction loop) EU FP6 integrated project,” http://chil.server.de/.

[HERMES] “HERMES (cognitive care and guidance for active aging) EU FP7 specific targeted research project,” http://www.fp7-hermes.eu.

[Patrikakis10] C. Patrikakis, A. Pnevmatikakis, P. Chippendale, M. Nunes, R. Santos Cruz, S. Poslad, W. Zhenchen, N. Papaoulakis, and P. Papageorgiou, “Direct your personal coverage of large athletic events,” Multimedia, IEEE, November 2010.

[Talantzis12] F. Talantzis, A. Pnevmatikakis, and A. G. Constantinides, Audio-Visual Person Tracking: A Practical Approach. Imperial College Press, 2012.

[Arulampalam02] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking,” IEEE Transactions on Signal Processing, vol. 50, no. 2, pp. 174–188, February 2002.

[Jaffre03] G. Jaffre and A. Crouzil, “Non-rigid object localization from color model using mean shift,” in IEEE International Conference on Image Processing (ICIP 2003), Barcelona, Spain, September 2003, pp. 317–320.

[Jones02] M. J. Jones and J. M. Rehg, “Statistical color models with application to skin detection,” International Journal of Computer Vision, vol. 46, no. 1, pp. 81–96, 2002.

[Babenko09] B. Babenko, M.-H. Yang, and S. Belongie, “Visual Tracking with Online Multiple Instance Learning,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2009), Miami Beach, FL, USA, June 2009.

[Perez04] P. Perez, J. Vermaak, and A. Blake, “Data fusion for visual tracking with particles,” Proceedings of IEEE, vol. 92, no. 3, pp. 495–513, 2004.

[Kitagawa96] G. Kitagawa, “Monte carlo filter and smoother for non-gaussian nonlinear state space models,” Journal of Computational and Graphical Statistics, vol. 5, no. 1, pp. 1–25, 1996.

[Pnevmatikakis13] A. Pnevmatikakis, A. Stergiou, N. Katsarakis and Th. Petsatodis, “Visual Measurement Cues for Face Tracking”, DSP2013, Santorini, Greece, July 2013, accepted for publication.

[Blackman99] S. Blackman and R. Popoli, “Design and Analysis of Modern Tracking Systems”, Artech House radar library, 1999.

[Buehren11] M. Buehren, “Functions for the rectangular assignment problem”, http://www.mathworks.com/matlabcentral/fileexchange/6543-functionsfor-the-rectangular-assignment-problem, 2011.

[OpenCVtracking] http://docs.opencv.org/trunk/modules/video/doc/motion_analysis_and_object_tracking.html#camshift