Stereo Based 3D Tracking and Scene Learning, employing Particle Filtering within EM

Trausti Kristjansson[1], Hagai Attias[1], and John Hershey[1]

Microsoft Research,
One Microsoft Way, Redmond 98052, USA
{traustik, hagaia, hershey}@microsoft.com
http://www.research.microsoft.com/users/traustik

Abstract. We present a generative probabilistic model for 3D scenes with stereo views. With this model, we track an object in 3 dimensions while simultaneously learning its appearance and the appearance of the background. By using a generative model for the scene, we are able to aggregate evidence over time. In addition, the probabilistic model naturally handles sources of variability.
For inference and learning in the model, we formulate an Expectation Maximization (EM) algorithm where Rao-Blackwellized Particle filtering is used in the E step. The use of stereo views of the scene is a strong source of disambiguating evidence and allows rapid convergence of the algorithm. The update equations have an appealing form and as a side result, we give a generative probabilistic interpretation for the Sum of Squared Differences (SSD) metric known from the field of Stereo Vision.

1 Introduction

We introduce a generative, top-down viewpoint for tracking and scene learning. We assume that a scene is composed of a moving object in front of a background. The scene model is shown in Figure 1(a). Within this paradigm, we can simultaneously learn the appearance of the background and the object, while the object moves in 3 dimension within the scene.

The algorithm is based on a probabilistic generative modelling approach. Such a model describes the scene components and the process by which they generate the observed data. Being probabilistic, the model can naturally describe the different sources of variability in the data. This approach provides a framework for learning and tracking, via the EM algorithm associated with the generative model. In the E-step, object position is inferred and sufficient statistics are computed; in the M-step, model parameters, including object and background appearances, are updated.

Sensor fusion is another important advantage of the probabilistic generative modelling approach. Whereas a bottom-up approach would process the signal from each camera separately, then combine them into an estimate of the object position, our approach processes the camera signals jointly and in a systematic fashion that derives from the model.

The use of a stereo view of the scene turns out to be of significant value over the use of a monocular view. It allows the algorithm to locate and track an object, even

when the prior model of the object appearance is uninformative e.g. when initialized to random values. As a consequence, only a small number of EM iterations are required for convergence.

In section 2 we discuss prior work and relate it to the current work. In section 3, we introduce the scene model. When the object moves within the scene, the connectivity of the graphical model for the scene changes. The connectivity is dictated by the geometry of the scene and is captured by the coordinate transformations that are discussed in section 4. In section 5 we discuss the Generalized EM algorithm, emphasizing the intuitive interpretation of the update equations of the E-step. Section 5.1 discusses the combination of EM and particle filtering for inferring the location and learning appearances. Results for a video sequence are given in section 6.

2 Related Work

The work presented here can be viewed as drawing on and bridging the fields of 3D tracking[1, 2], stereo vision[3] and 2-D scene modelling[4]. We briefly review related work in these fields and relate and contrast with the current work.

Tracking an object in tree dimensions is useful for a variety of applications [5, 6] ranging from robot navigation to human computer interfaces. Most tracking methods rely on a model of the object to be tracked. Object models are usually constructed by hand [7, 1, 2]. For example, Schodl et al. [2] use a textured 3D polygonal model and use gradient descent in a cost function.

Our model is similar to these methods in that we use an appearance map of the object, and track it in 3 dimensions. These methods rely on strong prior models in order to do tracking from a monocular view. As we use a stereo view of the scene, our method does not require prior hand construction of the model of the object, e.g. the face, and we are able to learn a model. Once a model has been learned, one can track the object using only a monocular view.

The objective of most stereo vision work has been to extract a depth map for an image. The evidence is in the form disparity between pixels in two or more views of the same scene [3, 8, 9]. Most stereo vision methods calculate a disparity cost based on this evidence, such as Sum of Squared Differences (SSD)[10]. In section 5.3 we offer a generative probabilistic interpretation for SSD.

Frey and Jojic [4, 11], and Dellaert et al.[12] use generative top-down models. They use layered 2D models, and learn 2D templates for objects that move across a background[13]. When using a monocular view from a single camera, learning the appearance of objects that can occlude each other is a hard problem. By incorporate stereo views of a scene, we can resolve the identifiability problem inherent with using a single camera and can more easily track an object in 3 dimensions.

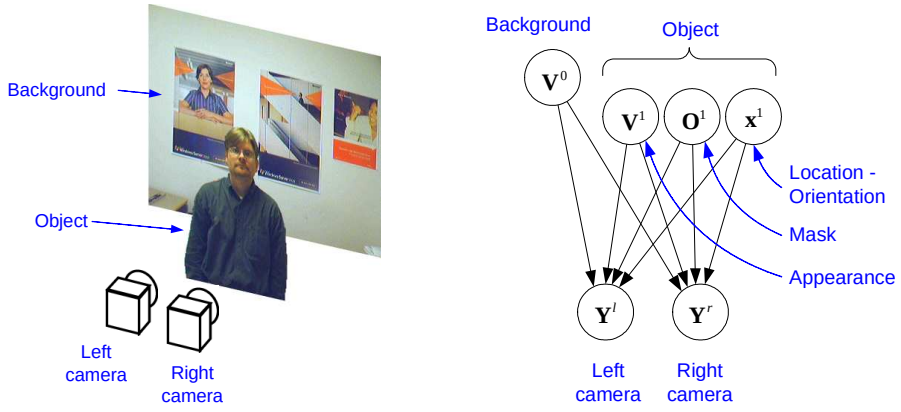
Recently, a great deal of attention has been paid to particle filtering in various guises. Blake et al.[14, 15] use models based on tracking spline outlines of objects[16]. Other researchers have extended this to appearance based models [17]. As with the tracking methods discussed before, the models are usually constructed by hand rather than learned.

We use Rao-Blackwellized[18] particle filtering to track the position and orientation of an object within a scene. In Rao Blackwellized particle filtering, the model contains random variables represented by parametric distributions as well as sampled random variables represented as particle sets. When performing inference over the sampled random variable, one must integrate over the parametric random variables.

We extend the standard Particle Filtering[14] paradigm in two ways. First, we use particle filtering in conjunction with stereo observations to track an object in 3 dimensions. Secondly, unlike most tracking paradigms, we are also able to learn the appearance of the objects in the scene, as they move in the scene [19, 18]. We believe this is the first demonstration of this algorithm for real data.

3 The Stereo Scene Model

The scene model is shown if Figure 1(a). The figure shows a background, a "cardboard cutout" object in front of the background that occludes part of it and two cameras. Figure 1(b) shows the equivalent graphical model. We assume that the object will be seen at different locations in the two cameras due to stereo disparity and that the cameras are aligned such that the same background image is seen in both cameras.



(a) Scene with stereo cameras.

(b) Generative model for stereo observations.

Fig. 1. (a)Schematic of scene with stereo cameras. (b) Generative model for stereo observations of a scene with a single object that partially occludes and a background.

In this graph, $\mathbf{V}^0$ is the background image, $\mathbf{V}^1$ is the object, $\mathbf{O}^1$ is the transparency mask of the object, $\mathbf{x}^1$ is a vector containing the position and orientation of the object, $\mathbf{Y}^l$ is the observed image in the left camera and $\mathbf{Y}^r$ is the observed image in the right camera.

The position variable $\mathbf{x}^1$ is a continuous random variable which contains at least 3 spacial coordinates of the object, allowing for 3D translation within the scene.

We use multivariate Gaussians with diagonal covariance matrices to model all appearances. Hence, the appearance model of the background is

$$p(\mathbf{V}^0) = \prod_j N(v_j^0; \mu_j^0, \eta_j^0), \tag{1}$$

where v_j^0 is the value of pixel j, μ_j^0 is the mean, and η_j^0 is the precision.

The model for the object contains three components: a template, a transparency mask and a position. Again, the appearance is modelled by a multivariate Gaussian with diagonal covariance matrix,

$$p(\mathbf{V}^1) = \prod_i N(v_i^1; \mu_i^1, \eta_i^1) \tag{2}$$

where v_i^1 is the value of pixel i, μ_i^1 is the mean, and η_i^1 is the precision.

Pixels in the object model can be opaque or transparent. We use discrete mixing, i.e. a pixel is either completely opaque or transparent. The prior distribution is

$$p(\mathbf{O}^1) = \prod_i \left[\alpha_i o_i + (1-\alpha_i)(1-o_i)\right]. \tag{3}$$

where o_i is the value of pixel i and α_i is the probability that the pixel is opaque.

The distribution for the position/orientation random variable, is handled differently from other variables in the model. It is represented by a particle set. A particle set is a set of vectors $\{x_s\}$ where each vector (also called a particle) represents a position of the object and each particle is associated with a weight $\{q(x_s)\}$.

We use a Gaussian for the prior for the position of the object

$$p(\mathbf{x}^1) = N(\mathbf{x}^1; \mu_x, \eta_x) \tag{4}$$

where μ_x is the mean and η_x is the precision. This is used when generating the initial set of particles and for recovering particles that land outside the a bounding volume.

When generating instances of the left and right camera images, we first sample from the background model, then we choose a position for the object and sample from the object appearance model. The appearance of the object is then overlayed on the background, for pixels where the object is opaque. For example, the value of the j-th pixel y_j^l in in the left image $\mathbf{Y}^l$ is

$$y_j^l = o^1_{\xi(x,j)} \cdot v^1_{\xi(x,j)} + (1 - o^1_{\xi(x,j)}) \cdot v_j^0 + \varepsilon^l \tag{5}$$

In words, pixel y_j^l takes the value of the object pixel $v^1_{\xi(j)}$ if it is opaque (i.e. $o^1_{\xi(j)} = 1$) or the value of the background v_j^0 if it is transparent. Finally we add Gaussian pixel noise ε^l with precision λ. Pixels in the right image are of course found similarly. The function $\xi(x, j)$ maps coordinates depending in the position of the object, and will be discussed in the next section. If we assume all variances are zero, the process of generating from this model is analogous to rendering the scene using standard computer graphics methods.

The prior distribution for a pixel in the left image y_j^l is

$$p(y_j^l|\mathbf{V}^0,\mathbf{V}^1,\mathbf{O}^1,\mathbf{x}) = \begin{cases} N(y_j^l; v^1_{\xi(x,j)}, \lambda) & \text{if } o^1_{\xi(x,j)} = 1 \\ N(y_j^l; v_j^0, \lambda) & \text{if } o^1_{\xi(x,j)} = 0 \end{cases} \tag{6}$$

The complete probability distribution for the sensor images is the product of the distributions for the individual pixels,

$$p(\mathbf{Y}^l,\mathbf{Y}^r|\mathbf{V}^0,\mathbf{V}^1,\mathbf{O}^1,\mathbf{x}) = \prod_j p(y_j^l|\mathbf{V}^0,\mathbf{V}^1,\mathbf{0}^1,\mathbf{x}) \cdot \prod_j p(y_j^r|\mathbf{V}^0,\mathbf{V}^1,\mathbf{O}^1,\mathbf{x}). \tag{7}$$

4 Coordinate transformations

The object can be at various locations and orientations. Hence, the mapping from coordinates on the object model to the image sensor will change. If the object is close to the camera, then each pixel on the object may map onto many pixels on the camera sensor, and if it is far away, many pixels map onto a single pixel in the camera sensor.

We define a set of functions that map between coordinates in the various appearance models we will be using. We assume that the cameras are pinhole cameras, looking along the negative z axis. For example, if the distance between the two cameras is 10 cm, then left eye is located at $[-5,0,0]^T$ and the right camera is at $[5,0,0]^T$. The mapping is defined in terms of transformations of homogeneous coordinates. Homogenous coordinates allow us perform translations and perspective projections in a consistent framework and are commonly used in computer graphics. A point in homogenous coordinates includes a 4th component h, i.e. (x,y,z,h). Assuming a flat object $\mathbf{V}^1$, the transformation from the matrix indices of the object into the matrix indices of the left sensor $\mathbf{Y}^l$, is denoted as $jl = \xi^{v\rightarrow yl}(x,i)$. This mapping is defined as

$$\begin{bmatrix} indx_i(\mathbf{Y}^l, jl) \\ indx_j(\mathbf{Y}^l, jl) \\ 0 \\ 1 \end{bmatrix} = \mathbf{SM} \cdot \mathbf{PRS}(\mathbf{x}) \cdot \mathbf{EYE}(l) \cdot \mathbf{W}(\mathbf{x}) \cdot \mathbf{MO} \cdot \begin{bmatrix} indx_i(\mathbf{V}^1, i) \\ indx_j(\mathbf{V}^1, i) \\ 0 \\ 1 \end{bmatrix} \tag{8}$$

where $indx_i(\mathbf{V}^1,i)$ denote the row index of pixel i in the object and $indx_j(\mathbf{V}^1,i)$ denotes the column index. Similarly, $indx_i(\mathbf{Y}^l,jl)$ denotes the row index of pixel jl in the left sensor image, and $indx_j(\mathbf{Y}^l,jl)$ denotes the column index. **MO** transforms from matrix-coordinates to canonical position in physical coordinates, $\mathbf{W}(\mathbf{x})$ transforms from canonical object position to the actual position $\mathbf{x}$ of the object in physical coordinates (relative to the camera coordinate system). $\mathbf{EYE}(l)$ is the transformation due to the position of the left eye. In our case, it is simply a shift of 5 for along x for the left camera, and -5 for the right camera. $\mathbf{PRS}(\mathbf{x})$ is the perspective projective transformation, which depends on the distance of the object from the camera. **SM** maps from physical sensor coordinates to sensor matrix coordinates.

To transform an observed image into the object, we map the matrix indices of the object through this transformation, round the result to the nearest integer, and then retrieve the the values of in the image matrix at those indices.

We will have a need for additional coordinate transformations: the inverse mapping of Eqn. (8) is $i = \xi^{yl \rightarrow v}(x, jl)$ which maps left sensor coordinates into object model coordinates. The function $jr = \xi^{v \rightarrow yr}(x, i)$ maps from the coordinates of the object model into the left sensor matrix, and the inverse transformation is $i = \xi^{yr \rightarrow v}(x, jr)$

An interesting consequence of using stereo cameras and working in world coordinates is that coordinates have a physical meaning. For example, the matrix **MO** defines the physical resolution of the object appearance model. In our experiments, the physical size of one pixel on the surface of the object is about $1\ cm \times 1\ cm$. If only a single camera is used, it is not be possible to determine the scale at which an object should be modelled.

5 EM-PF Algorithm for Learning Stereo Scenes

Now we present an EM algorithm, that employs Rao Blackwellized particle filtering to compute approximations to the model posteriors in an approximate E-step.

We employ two types of approximations to compute the model posteriors in the E step of the algorithm. The first approximation comes from the factorization of the graph, and the second from the approximation of the location posterior with a particle set.

5.1 The EM - Particle Filtering Hybrid Algorithm

The graph in Figure 1(b) hides the fact that the connectivity of the graph changes depending on the position $\mathbf{x}^1$ of the object. Each pixel in the object $\mathbf{V}^1$ can be connected to any pixel in $\mathbf{Y}$, depending on the position $\mathbf{x}^1$. Another way of viewing this is that *every* pixel in the object connects to *every* pixel in the image, and the position of the object determines which edges are "turned on". Thus the graph is hugely loopy. Once a position has been chosen, the connectivity of the graph is dramatically reduced[1].

It is problematic to use a parametric distribution for the position variable x since we need to integrate over it which entails integrating over discrete topologies of the graph. This is the motivation for representing the location variable with a particle set, and using particle filtering for inferring posteriors for the location variable x. Algorithm 1 shows the hybrid EM - particle filtering algorithm, for stereo scene analysis.

When learning, we start by sampling from a location prior, and initializing the parameters of the background and object models to random values. In the E step, we compute posterior distributions for the appearance models, and weights for each location particle. When going to the next frame, we re-sample the particles based on those weights and the particles are then passed through a dynamic distribution. The M step is performed after going through the whole sequence of frames.

Various extensions of the basic particle filtering algorithm are possible, e.g. that use proposal distributions[15] or iterative updates within each frame[16] to get a more representative particle set for the location.

[1] The graph still has "horizontal" chains, which we will discuss in the next section.

Algorithm 1 EM - Particle filtering hybrid algorithm

Initialize model parameters μ_0, η_0, μ^1, η^1, α^1.
for $nGEM = 1$ to $num\_GEM\_iterations$ **do**

Approximate E step
Sample particle set $\{x\}_0$ from location prior $p(x)$.
for $f = 1$ to num_frames **do**
$\{x\}'_f \leftarrow sample(p(x_{s,f}|x_{s,f-1}))$ – send particles through dynamic distribution
Estimate parameters of approximate posteriors $\overline{\alpha}^1$, $\overline{\eta}^1$, $\overline{\mu}^1$, $\overline{\eta}_0$ and $\overline{\mu}_0$
Calculate particle weights $q(x_s)$
$\{x\}_f \leftarrow resample(\{x\}'_f, \{q(x_s)\})$ – re-sample particles based on weights
end for

M step
Update model parameters μ_0, η_0, μ^1 η^1 and α^1
end for

5.2 Graph Factorization

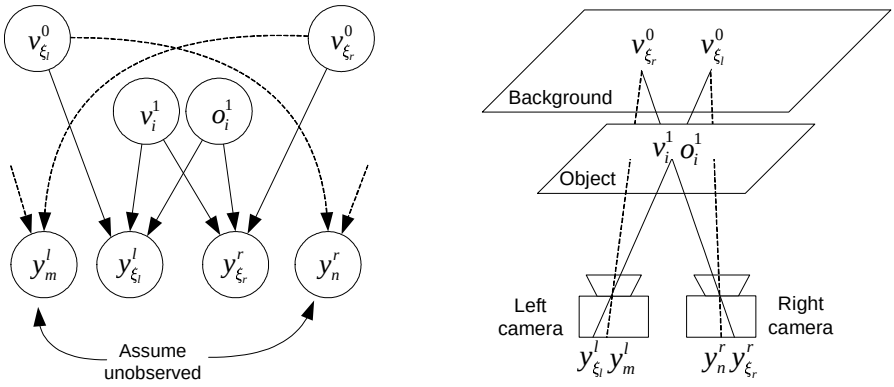
For a particular setting of the position variable **x**, the original graph factors into chains along the epipolar lines. In other words, the posterior distribution of a pixel in the object is not only dependent on the directly observed pixels it impinges on, but also depends indirectly on a large number of other pixels along the same epipolar line. Part of such a graph is shown in Figure 2. In order to make inference efficient we would like to factor the model and omit the dependence on pixels that are not directly observed[2]. This can be accomplished by assuming that only the directly observed pixels in the camera sensors are observed and all other pixels are unobserved. This has the effect of decoupling the graph and leads to an approximation for the true posterior. From the perspective of inference, assuming that neighboring pixels are unobserved is equivalent to allowing those pixels to take on any values, including the values actually observed.

5.3 Posterior distributions of E-step

We now turn our attention to the posterior distributions for the object model $\mathbf{V}^1$ and the position x. These distributions are required in the E step of the learning algorithm. We omit discussion for the posterior distributions for the background and mask due to space constraints as they are intuitively analogous.

The manifestation of stereo in the equations below is one of the more important and pleasing result of this paper. Terms that can be interpreted as "appearance" terms as well, as "stereo" terms, fall out naturally from the generative model without any ad-hoc combination of these concepts.

[2] We also experimented with variational inference. Using variational inference, we were unable to learn the parameters of the occlusion variables. We believe this is due to the omission of important dependence structure, which the mean field approximation ignores.



(a) Portion of epipolar graph. (b) Geometric interpretation

Fig. 2. (a) Portion of graphical model corresponding to an epipolar line. Notice that once the observation nodes have been set, the posterior of the random variable o_i^1 is dependent on the directly observed pixels $y^l_{\xi_l}$ and $y^r_{\xi_r}$ *and* on y^l_m and y^r_n (and so on), which are not directly observed. The dependence comes from their influence on the background nodes $v^0_{\xi_l}$ and $v^0_{\xi_r}$. In Section 5.2 we describe how the chain is factored.(b) Geometric interpretation of graph in (a).

Posterior for object V^1 By assuming only the directly observable pixels in the sensors are observed, the posterior associated with pixel i in the object becomes

$$p(v_i^1, o_i^1, v^0_{\xi l}, v^0_{\xi r} | x, y^l_{\xi l}, y^r_{\xi r}) \tag{9}$$

$$\propto p(v_i^1, o_i^1, v^0_{\xi l}, v^0_{\xi r}, x, y^l_{\xi l}, y^r_{\xi r}) \tag{10}$$

$$= \begin{cases} p(o_i^1 = 1) p(y^l_{\xi l} | v_i^1, x) p(y^r_{\xi r} | v_i^1, x) p(v^0_{\xi l}) p(v^0_{\xi r}) p(v_i^1) p(x) & \text{if } o_i^1 = 1 \\ p(o_i^1 = 0) p(y^l_{\xi l} | v^0_{\xi l}, x) p(y^r_{\xi r} | v^0_{\xi r}, x) p(v^0_{\xi l}) p(v^0_{\xi r}) p(v_i^1) p(x) & \text{if } o_i^1 = 0 \end{cases} \tag{11}$$

To get the posteriors over the pixels of the object, we marginalize out o_i^1, $v^0_{\xi l}$ and $v^0_{\xi r}$. The posterior for v_i^1, given a location and the sensor images is a mixture of two Gaussians

$$p(v_i^1 | x, y^l_{\xi l}, y^r_{\xi r}) = c\alpha_i^1 w_1 N(v_i^1, \mu_{observed}, \eta_{observed}) \tag{12}$$

$$+ c(1 - \alpha_i^1) w_0 N(v_i^1, \mu_{not\ observed}, \eta_{not\ observed}) \tag{13}$$

where c is a normalizing constant. α_i^1 is the prior for the mask variable, and w_1 and w_0 are the mixture weights.

This is a very intuitive result. The first mixture is for the case that the mask is opaque for that pixel, and the second mixture is for the case that it is transparent. The mode of

the "opaque" component is

$$\mu_{observed} = \frac{1}{\eta_i^1 + \lambda^l + \lambda^r}\left[\eta_i^1\mu_i^1 + \lambda^l y_{\xi^l}^l + \lambda^r y_{\xi^r}^r\right] \tag{14}$$

which is a weighted average of what is observed, and the prior mode η_i^1. The weight w_1 for this component is composed of two Gaussian factors

$$w_1 = N(y_{\xi^l}^l - y_{\xi^r}^r; 0, \frac{\lambda^l\lambda^r}{\lambda^l+\lambda^r}) \cdot N(\frac{1}{\lambda^l+\lambda^r}\left[\lambda^l y_{\xi^l}^l + \lambda^r y_{\xi^r}^r\right]; \mu_i^1, \frac{(\lambda^l+\lambda^r)\eta_i^1}{\lambda^l+\lambda^r+\eta_i^1}). \tag{15}$$

The first factor is the "stereo" factor, which is maximized when there is a close correspondence between what is seen in the left and right images i.e. $y_{\xi^l}^l = y_{\xi^r}^r$, and the second factor, the "appearance" factor, is maximized when the prior for the object appearance μ_i^1 matches the (weighted) mean observation. Hence the weight will be large for cases when there is good stereo correspondence and the observation matches the prior.

The second component in the posterior in Eqn.(12) is for the case when the mask is transparent. In this case the mixture component is just equal to the prior. The weight w_0 for this component contains two factors that can be thought of as measuring the evidence that the observed pixel came from the background.

$$w_0 = N(y_{\xi^l}^l, \mu_{\xi^l}^0, \frac{\eta_{\xi^l}^0\lambda^l}{\eta_{\xi^l}^0+\lambda^l}) \cdot N(y_{\xi^r}^r, \mu_{\xi^r}^0, \frac{\eta_{\xi^r}^0\lambda^r}{\eta_{\xi^r}^0+\lambda^r}) \tag{16}$$

The first term is maximized when the observation matches the left background pixel, and the second term is maximized the right background pixel matches the observed pixel in the right camera

Notice that Equation (12) is for a particular position of the the object. The approximate posterior for the object appearance, can now be written as a Gaussian mixture model with a large number of mixtures. In fact it will have $2 \cdot nsamp$ mixtures, where $nsamp$ is the number of particles in $\{x_s\}$. The weight of each mixtures is the particle weight $q(x_s)$. Hence, the posterior of the object appearance is

$$q(v_i^1|y_{\xi^l}^l, y_{\xi^r}^r) = \sum_{x_s} q(x_s)p(v_i^1|x_s, y_{\xi^l}^l, y_{\xi^r}^r). \tag{17}$$

5.4 Posterior for x

The posterior for the position variable x is represented by the particle set $\{x_s\}$ and associated weights $\{q(x_s)\}$. The posterior distribution for the position x can be approximated at the position of the particles x_s as

$$p(x_s|\mathbf{Y}^l, \mathbf{Y}^r) \approx q(x_s) = \frac{p(x_s, \mathbf{Y}^l, \mathbf{Y}^r)}{\sum_k p(x_k, \mathbf{Y}^l, \mathbf{Y}^r)}. \tag{18}$$

To arrive at an expression for the weight of a particle, we need to integrate over all parametric distributions (Rao-Blackwellization). By doing so, $p(x_s, \mathbf{Y}^l, \mathbf{Y}^r)$ can be shown to

be

$$p(x_s, \mathbf{Y}^l, \mathbf{Y}^r) = \prod_i \int p(x_s, \mathbf{Y}^l, \mathbf{Y}^r, v_i^1, \mathbf{V}^0, \mathbf{O}^1) dv_i^1 d\mathbf{V}^0 d\mathbf{O}^1$$
$$= \prod_i \left[\alpha_i^1 w_1(i) + (1-\alpha_i^1) w_0(i)\right] \quad (19)$$

where α_i^1, $w_0(i)$ and $w_1(i)$ were defined above.

5.5 Generative Probabilistic Interpretation of SSD

The Sum of Squared Differences (SSD) metric is commonly used in stereo vision [3, 10] to measure how well a patch in one image matches a patch from another image, as a function of disparity. It is interesting to note that SSD can be seen as a component or special case of Equation (18).

Equation (18) gives the posterior distribution $p(x_s|\mathbf{Y}^l, \mathbf{Y}^r)$ for the location of the object and can be interpreted as measuring the "fit" of the hypothesized position to the observed data. Recall that Equation (18) contains both "appearance" related terms and "stereo" related terms.

To see the relationship of Equation (18) to the SSD metric, we assume that the appearance model is completely uninformative ($\eta^i = 0$), that the object is completely opaque ($\alpha_i^1 = 1$ for all i), and take the log to arrive at the form

$$\log(p(x|\mathbf{Y}^l, \mathbf{Y}^r)) \propto \log\left(\prod_i \left[\alpha_i^1 w_1(i) + (1-\alpha_i^1) w_0(i)\right]\right) = \sum_i \log w_1(i). \quad (20)$$

Recall that the first term in the weight w_1 is $N(y^l_{\xi^l(x,i)} - y^r_{\xi^r(x,i)}; 0, \frac{\lambda^l \lambda^r}{\lambda^l + \lambda^r})$. Hence, for this special case

$$\log(p(x|\mathbf{Y}^l, \mathbf{Y}^r)) \propto \sum_i (y^l_{\xi^l(x,i)} - y^r_{\xi^r(x,i)})^2 \quad (21)$$

which is exactly equivalent to the SSD over the whole image.

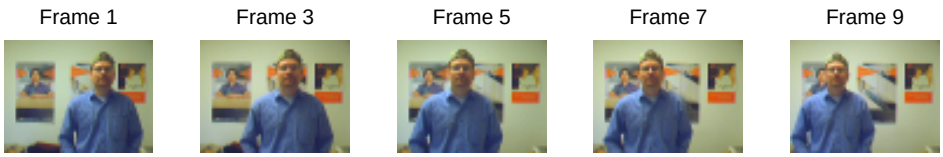


Fig. 3. Training data consists of a short sequence of 10 stereo video frames. The frames were down sampled to 64x48 pixels. The figure shows the frames from the left camera. Notice that the person approaches the camera from the the right and then recedes to the left. The trajectory is shown in Figure 6.

6 Experiments

A short video was recorded using a stereo video camera. The frame rate was 2 frames per second. A subset of the 10 frame sequence used to train the model is shown in Figure 3. Notice that the person approaches the camera from the the right and then recedes to the left.

Figures 4. and 5. shows the models that were found as a result of running the algorithm on the 10 frames shown in Figure 3. In these experiments, 500 particles were used. As can be seen in Figure 4., the background image is learned precisely in most areas. However, in areas where the background is never seen, the background has not been learned, and the variance is high.

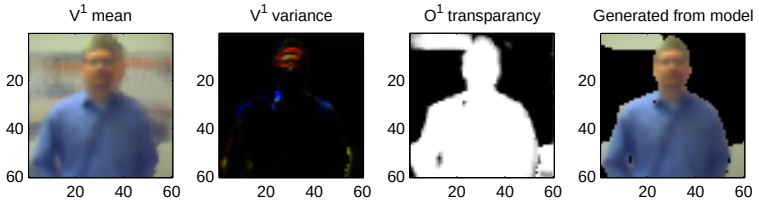


Fig. 4. Model learned for object. The model is comprised of an Gaussian appearance model $\mathbf{V}^1$ and a discrete transparency model $\mathbf{O}^1$. The leftmost figure shows the mean of $\mathbf{V}^1$, the second figure shows the variance of $\mathbf{V}^1$ (notice higher variance on the forehead). The third plot shows the probability of each pixel being opaque. The rightmost plot shows an object generated from the model. Notice that patches of the background where there is no detail, have been associated with the object.

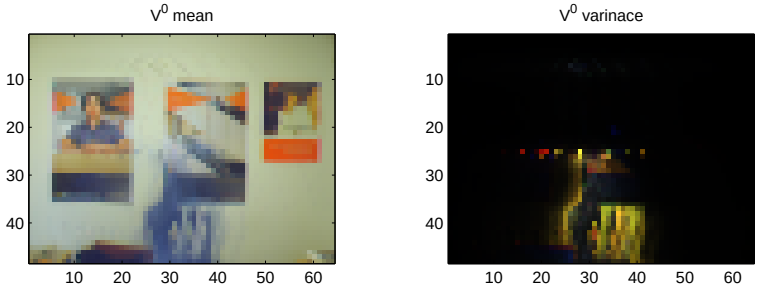


Fig. 5. Model learned for background. The model $\mathbf{V}^0$ is a multivariate gaussian. The left images shows the means, and the right image shows the variances of $\mathbf{V}^0$. Notice that the areas where the background was never observed remain the color of the object and have high variance

The transparency mask has been learned well, except in areas where there is no texture in the background which would allow the model to disambiguate these pixels.

Notice that the appearance model has been learned quite well. As can be seen in Figure 3., highlights and specularity of the forehead, nose and shirt vary between frames. The consequence of this is the large variance in these areas. A second factor that introduces variance is that the model assumes the object is flat. Hence, there will be distortion due to the different perspectives of the two cameras. The model allows for this discrepancy by assigning larger variance to the object appearance model along the edges of the face. A third source of variability comes from the inference algorithm itself. The sampling resolution can be too coarse, which prevents the algorithm from accurately finding the mode of the location posterior. This does not seem to be a problem here. This effect can be reduced in a number of ways, including increasing the number of particles and using higher order dynamics in the temporal distribution.

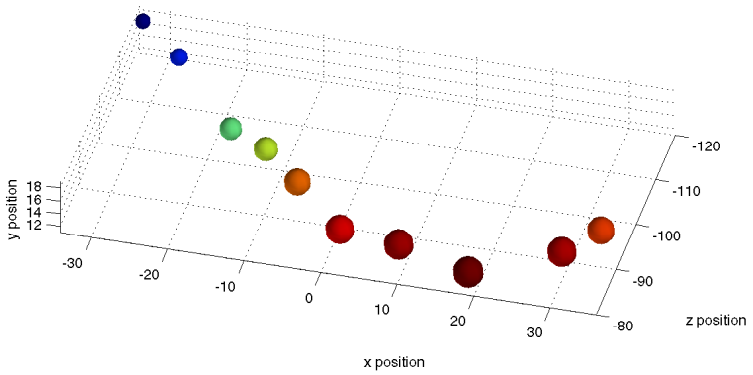


Fig. 6. The mode of the location distribution in iteration 9 of the EM algorithm. The units are approximately centimeters. Notice that there is considerable variation in both depth and horizontal location.

Figure 6. shows the trajectory of the mode of the distribution for the location variable x. The figure clearly shows a right to left trajectory of the person that starts in the right hand side of the frame, moves closer and to the center and then recedes to the left.

7 Discussion

The algorithm requires a large number of coordinate transformations as well as evaluations of posteriors for the transformed images. The complexity of the algorithm is $O((m+n) \cdot it \cdot nsamp \cdot fr)$ where m is the number of pixels in the background, n is the number of pixels in the object model, it is the number of iterations of GEM, $nsamp$ is the number of samples and fr is the number of frames[3].

The transformations required for inference and learning resemble those used in computer graphics. Commodity 3-D graphics accelerators are capable of performing

[3] Each frame takes about 15 seconds on a 2.8GHz Pentium running Matlab code.

the required computations at high speeds and we anticipate that a fast implementations can be achieved.

In some cases, it is a poor assumption that the background is at a relatively large distance and can be modelled as a planar surface. This can happen when there are stationary objects in the scene at a similar distance as the object we wish to model and track. For this case it may be advantageous to use separate background models for the two cameras.

Particle Filtering and other Markov Chain Monte Carlo methods are considered slow techniques. In addition, when a generative top-down model is used, exact inference will theoretically require the search over a huge space of possible configurations of the hidden model. With continuous location variable, this space is in fact infinite. Despite this, we are able to both track and learn the appearance of the objects in a scene. This is partly due to the advantageous prior structure imposed by the top-down model, partly due to the strong disambiguating information provided by stereo views of the scene and partly due to an inference algorithm that is able to search only over the regions of the hidden variable space that are likely to contain the best explanation for the visible scene.

Stereo information allows the algorithm to latch on to the correct position of the object immediately, even when the appearance model is of no help e.g. when it is initialized to random values. Hence, stereo information allows the algorithm to track and learn appearance without any prior knowledge of the appearance of an object.

When we applied an equivalent monocular algorithm using a single camera to the above data, the algorithm did not track the object, did not learn the object model and consistently fell into local minima. However, once an appearance model has been learned (using stereo) one can switch to using a single camera to track the object.

A strength of generative probabilistic modes is the consistent fusion of multiple types of information where noise and uncertainty are correctly taken into account. In the current paper, we fuse appearance, stereo views and views through time to learn a single underlying representation that explains a scene. Information from multiple frames is automatically used to fill in portions the model that are only observed in a subset of frames.

The framework uses a (simple) generative 3D model of a scene and we show that we can successfully perform inference in such a top-down model. In contrast, the majority of methods in computer vision are bottom up methods. Aggregation of multiple sub models into larger models is a challenge for such approaches. Hence, we believe the extension of the current paradigm to be a very fruitful direction of further research, especially when it is desirable to construct consistent 3D representations of the world.

References

1. M. Malciu, F.P.: A robust model-based approach for 3d head tracking in video sequences. In: Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (FG'2000), Grenoble, France. Volume 1. (2000) 169–174
2. Schodl, I., Haro, A.: Head tracking using a textured polygonal model. In: In Proceedings of Workshop on Perceptual User Interfaces. (1998)

3. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International Journal of Computer Vision (2002) 7–42
4. Frey, B., Jojic, N.: Transformation-invariant clustering and dimensionality reduction using em. IEEE Transactions on Pattern Analysis and Machine Intelligence (2000)
5. Papanikolopoulos, N., Khosla, P., Kanade, T.: Vision and control techniques for robotic visual tracking. In: In Proc. IEEE Int. Conf. Robotics and Autmation. Volume 1. (1991) 851–856
6. Toyama, K.: Prolegomena for robust face tracking. Technical Report MSR Technical Report, MSR-TR-98-65, Microsoft Research (1998)
7. Jebara, T., Azarbeyejani, A., Pentland, A.: 3d structure from 2d motion. IEEE Signal Processing Magazine **16** (1999)
8. Sun, J., Shum, H.Y., Zheng, N.N.: Stereo matching using belief propagation. European Conference on Computer Vision (2002) 510–524
9. Scharstein, D., Szeliski, R.: Stereo matching with non-linear diffusion. Proc. of IEEE conferenc on Computer Vision and Pattern Recognition (1996) 343–350
10. Kanade, T., Okutomi, M.: A stereo matching algorithm with an adaptive window: Theory and experiment. IEEE Transactions on Pattern Analysis and Machine Intelligence **16** (1994) 920–932
11. Frey, B.J., Jojic, N.: Learning graphical models of images, videos and their spatial transformations. In: Proceedings of the Sixteenth Conference on Uncertainty in Artifical Intelligence. (2000)
12. Dellaert, F., Thrun, S., Thorpe, C.: Jacobian images of super-resolved texture maps for model-based motion estimation and tracking. In: IEEE Workshop on Applications of Computer Vision. (1998) 2–7
13. Wang, J., Adelson, E.: Representing moving images with layers. IEEE Transactions on Image Processing, Special Issue: Image Sequence Compression **4** (1994) 625–638
14. Blake, A., Isard, M.: Active Contours. Springer-Verlag (1998)
15. Isard, M., Blake, A.: Icondensation: Unifying low-level and high-level tracking in a stochastic framework. In: Proc. 5th European Conf. Computer Vision. Volume 1. (1998) 893–908
16. Kristjansson, T., Frey, B.: Keeping flexible active contours on track using metropolis updates. Advances is Neural Information Processing (NIPS) (2000) 859–865
17. Cootes, T., Edwards, G., Taylor, C.: Active appearance models. In: Proceedings of the European conference on Computer Vision. Volume 2. (1998) 484–498
18. Murphy, K., Russell, S.: Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. In: (Sequential Monte Carlo Methods in Practice)
19. Doucet, A., de Freitas, N., Murphy, K., Russell, S.: Rao-blackwellised particle filtering for dynamic bayesian networks. In: Proc. of Uncertainty in AI). (2000)