

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

Trausti Kristjansson¹, Hagai Attias¹, and John Hershey¹

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

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) cost 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 process the camera signal 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 the 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 map¹ 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 5.2 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 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 particle filtering to track the position and orientation of an object within a scene. We extend the basic 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 [18, 19]. We believe this is the first demonstration of this algorithm for real data.

3 The Stereo Scene Model

The scene model is shown in 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.

¹ In the current work, the object model is flat.

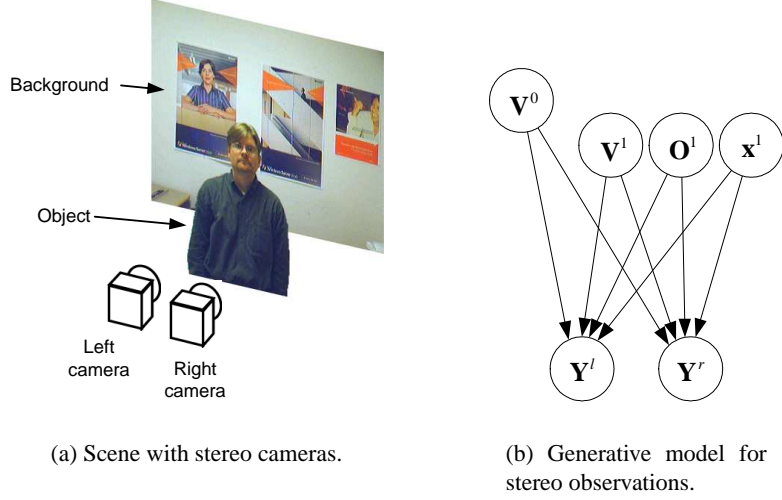


Fig. 1. (a) Schematic of scene with stereo cameras. (b) Generative model for stereo observations of a scene with a single object that partially occludes and a background.

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

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

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,

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

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. either a pixel is either completely opaque or transparent. The prior distribution is

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

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

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

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 overlaid 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

$$y_j^l = o_{\xi(x,j)}^1 \cdot v_{\xi(x,j)}^1 + (1 - o_{\xi(x,j)}^1) \cdot v_j^0 + \varepsilon^l \quad (5)$$

In words, pixel y_j^l takes the value of the object pixel $v_{\xi(j)}^1$ if it is opaque (i.e. $o_{\xi(j)}^1 = 1$) 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

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

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

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

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. Homogeneous coordinates allow us perform translations and perspective projections in a consistent framework and are commonly used in computer graphics. A point in homogeneous 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 $jl = \xi^{v \rightarrow y^l}(x, i)$. This mapping is defined as

$$\begin{bmatrix} \text{indx}_i(\mathbf{Y}^l, jl) \\ \text{indx}_j(\mathbf{Y}^l, jl) \\ 0 \\ 1 \end{bmatrix} = \mathbf{SM} \cdot \mathbf{PRS}(\mathbf{x}) \cdot \mathbf{EYE}(l) \cdot \mathbf{W}(\mathbf{x}) \cdot \mathbf{MO} \cdot \begin{bmatrix} \text{indx}_i(\mathbf{V}^1, i) \\ \text{indx}_j(\mathbf{V}^1, i) \\ 0 \\ 1 \end{bmatrix} \quad (8)$$

where $\text{indx}_i(\mathbf{V}^1, i)$ denote the row index of pixel i in the object and $\text{indx}_j(\mathbf{V}^1, i)$ denotes the column index. Similarly, $\text{indx}_i(\mathbf{Y}^l, jl)$ denotes the row index of pixel jl in the left sensor image, and $\text{indx}_j(\mathbf{Y}^l, jl)$ denotes the column index. \mathbf{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. \mathbf{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 $i = \xi^{y^l \rightarrow v}(x, jl)$ which maps left sensor coordinates into object model coordinates. The function $jr = \xi^{v \rightarrow y^r}(x, i)$ maps from the coordinates of the object model into the left sensor matrix, and the inverse transformation is $i = \xi^{y^r \rightarrow v}(x, jr)$

An interesting consequence of using stereo cameras and working in world coordinates is that coordinates have a physical meaning. For example, the matrix \mathbf{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 $1 \text{ cm} \times 1 \text{ cm}$. If only a single camera is used, it is not be possible to determine the scale at which an object should be modelled.

5 P-EM 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 reduced².

Algorithm 1 EM - Particle filtering hybrid algorithm

```

Initialize model parameters  $\mu_0, \eta_0, \mu^1, \eta^1, \alpha^1$ .
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  $\bar{\alpha}^1, \bar{\eta}^1, \bar{\mu}^1, \bar{\eta}_0$  and  $\bar{\mu}_0$ 
    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  $\mu_0, \eta_0, \mu^1, \eta^1$  and  $\alpha^1$ 
end for

```

Using a parametric distribution for the position variable is problematic since we need to integrate over x 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.

5.2 Graph Factorization

For a particular setting of the position variable \mathbf{x} , the original graph factors into chains along the horizontal dimension of the images. In other words, the posterior distribution

² The graph still has “horizontal” chains, which we will discuss in the next section.

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. In order to make inference efficient we would like to factor the model and omit the dependence on pixels that are not directly observed. This can be accomplished by assuming that only the directly observed pixels in the camera sensors are observed and all other pixels are unobserved³. 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.

We now turn our attention to the posterior distributions for the object model \mathbf{V}^1 and the position \mathbf{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 analogous⁴.

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

$$p(v_i^1, o_i^1, v_{\xi_l}^0, v_{\xi_r}^0 | x, y_{\xi_l}^l, y_{\xi_r}^r) \quad (9)$$

$$\propto p(v_i^1, o_i^1, v_{\xi_l}^0, v_{\xi_r}^0, x, y_{\xi_l}^l, y_{\xi_r}^r) \quad (10)$$

$$= \begin{cases} p(o_i^1 = 1) p(y_{\xi_l}^l | v_i^1, x) p(y_{\xi_r}^r | v_i^1, x) p(v_{\xi_l}^0) p(v_{\xi_r}^0) p(v_i^1) p(x) & \text{if } o_i^1 = 1 \\ p(o_i^1 = 0) p(y_{\xi_l}^l | v_{\xi_l}^0, x) p(y_{\xi_r}^r | v_{\xi_r}^0, x) p(v_{\xi_l}^0) p(v_{\xi_r}^0) p(v_i^1) p(x) & \text{if } o_i^1 = 0 \end{cases} \quad (11)$$

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

$$p(v_i^1 | x, y_{\xi_l}^l, y_{\xi_r}^r) = c \alpha_i^1 w_1 N(v_i^1, \mu_{observed}, \eta_{observed}) \quad (12)$$

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

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

$$\mu_{observed} = \frac{1}{\eta_i^1 + \lambda^l + \lambda^r} \left[\eta_i^1 \mu_i^1 + \lambda^l y_{\xi_l}^l + \lambda^r y_{\xi_r}^r \right] \quad (14)$$

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

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

³ This viewpoint has the flavor of Iterated Conditional Modes(ICM) [20]. In our case however, we will decouple the graph by assuming observed sensor nodes are unobserved.

⁴ The posterior distributions for the mask and background are given in Technical Report XXXX

The first factor 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 is maximized when the prior for the object appearance μ_i^l 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.

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

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 x_s . The weight of each mixtures is the particle weight $q(x_s)$. Hence, the posterior of the object appearance is

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

5.3 Posterior for \mathbf{x}

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

$$p(\mathbf{x}_s | \mathbf{Y}^l, \mathbf{Y}^r) \approx q(\mathbf{x}_s) = \frac{p(\mathbf{x}_s, \mathbf{Y}^l, \mathbf{Y}^r)}{\sum_k p(\mathbf{x}_k, \mathbf{Y}^l, \mathbf{Y}^r)}. \quad (18)$$

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

$$\begin{aligned} p(\mathbf{x}_s, \mathbf{Y}^l, \mathbf{Y}^r) &= \prod_i \int p(\mathbf{x}_s, \mathbf{Y}^l, \mathbf{Y}^r, v_i^1, \mathbf{V}^0, \mathbf{O}^1) dv_i^1 d\mathbf{V}^0 d\mathbf{O}^1 \\ &= \prod_i [\alpha_i^1 w_1(i) + (1 - \alpha_i^1) w_0(i)] \end{aligned} \quad (19)$$

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

5.4 Generative Probabilistic interpretation of SSD

The posterior distribution $p(\mathbf{x}, \mathbf{Y}^l, \mathbf{Y}^r)$ for the location of the object can be interpreted as measuring the “fit” of the hypothesized position to the observed data. To see the relationship of this posterior with the Sum of Squared Differences (SSD) cost commonly used in stereo vision [10, 3] we assume that the object is completely opaque ($\alpha_i^l = 1$ for all i), that the prior model for the appearance is completely uninformative ($\eta^l = 0$), and take the log to arrive at the form

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

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

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

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

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 2.

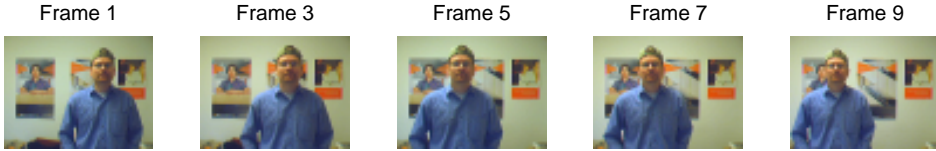


Fig. 2. Training data consists of a short sequence of 10 stereo video frames. The frames were down sampled to 64x48 pixels. The figure shows the frames from the left camera.

Figures 3. and 4. shows the models that were found as a result of running the algorithm on the 10 frames shown in Figure 2. In these experiments, 500 particles were used.

As can be seen in Figure 3., 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.

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 2., highlights and specularities of the forehead, nose and shirt vary between frames. The

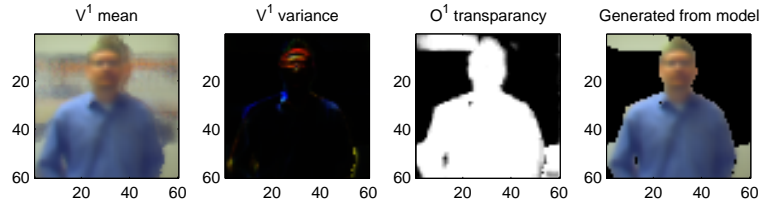


Fig. 3. Model learned for object. The model is comprised of an gaussian appearance model \mathbf{V}^1 and a discrete transparency model \mathbf{O}^1 . The leftmost figure shows the mean of \mathbf{V}^1 , the second figure shows the variance of \mathbf{V}^1 (notice higher variance on the forehead). The 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.

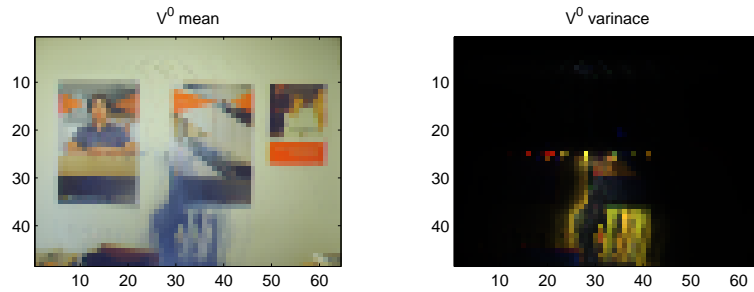


Fig. 4. Model learned for background. The model \mathbf{V}^0 is a multivariate gaussian. The left images shows the means, and the right image shows the variances of \mathbf{V}^0 . Notice how the areas where the object was modelled are accurately modelled and have low variance. The areas where the background was never observed, remain the color of the object.

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.

Figure 5. 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.

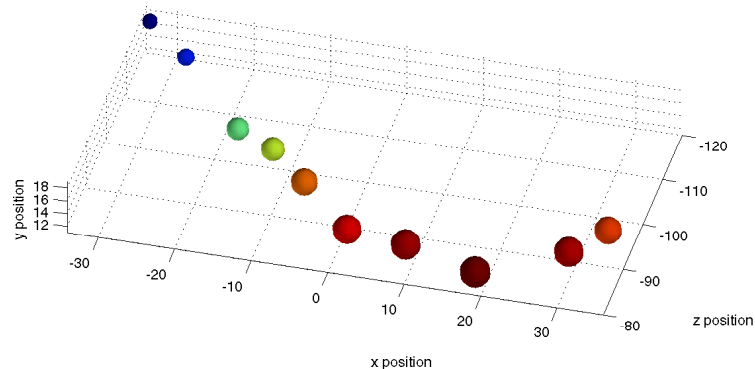


Fig. 5. The mode of the location distribution in iteration 9 of the EM algorithm. The units are approximately centimeters. Notice that there is considerable variation in both depth and horizontal location.

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 $O((m+n) \cdot it \cdot nsamp)$ 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 and $nsamp$ is the number of samples.

The transformations required for inference and learning resemble those used in computer graphics. Commodity 3-D graphics accelerators are capable of performing the required computations at high speed and we anticipate that 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 the stereo views of the scene as well as 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.

We have presented a generative probabilistic model for simple scenes, and shown how object can be tracked in three dimensions, while simultaneously learning the appearances of the objects within the scene.

References

1. F. Prêteux M. Malciu, "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*, March 2000, vol. 1, pp. 169–174.
2. I. Schodl and A. Haro, "Head tracking using a textured polygonal model," in *In Proceedings of Workshop on Perceptual User Interfaces*, November 1998.
3. D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, , no. 47, pp. 7–42, 2002.
4. B.J. Frey and N. Jojic, "Transformation-invariant clustering and dimensionality reduction using em," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.
5. N. Papanikolopoulos, P. Khosla, and T. Kanade, "Vision and control techniques for robotic visual tracking," in *In Proc. IEEE Int. Conf. Robotics and Automation*, 1991, vol. 1, pp. 851–856.
6. Kentaro Toyama, "Prolegomena for robust face tracking," Tech. Rep. MSR Technical Report, MSR-TR-98-65, Microsoft Research, 1998.
7. Tony Jebara, Ali Azarbeyejani, and Alex Pentland, "3d structure from 2d motion," *IEEE Signal Processing Magazine*, vol. 16, no. 3, 1999.
8. J. Sun, H-Y. Shum, and N-N. Zheng, "Stereo matching using belief propagation," *European Conference on Computer Vision*, pp. 510–524, 2002.
9. Daniel Scharstein and Richard Szeliski, "Stereo matching with non-linear diffusion," *Proc. of IEEE conferenc on Computer Vision and Pattern Recognition*, pp. 343–350, 1996.
10. Takeo Kanade and Masatoshi Okutomi, "A stereo matching algorithm with an adaptive window: Theory and experiment," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 9, pp. 920–932, 1994.
11. Brendan J. Frey and Nebojsa Jojic, "Learning graphical models of images, videos and their spatial transformations," in *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 2000.
12. Frank Dellaert, Sebastian Thrun, and Chuck Thorpe, "Jacobian images of super-resolved texture maps for model-based motion estimation and tracking," in *IEEE Workshop on Applications of Computer Vision*, October 1998, pp. 2–7.
13. J.Y.A. Wang and E.H. Adelson, "Representing moving images with layers.," *IEEE Transactions on Image Processing, Special Issue: Image Sequence Compression*, vol. 4, no. 5, pp. 625–638, 1994.
14. A. Blake and M. Isard, *Active Contours*, Springer-Verlag, 1998.
15. M. Isard and A. Blake, "Icondensation: Unifying low-level and high-level tracking in a stochastic framework," in *Proc. 5th European Conf. Computer Vision*, 1998, vol. 1, pp. 893–908.
16. T. Kristjansson and B.J. Frey, "Keeping flexible active contours on track using metropolis updates," *Advances in Neural Information Processing (NIPS)*, , no. 13, pp. 859–865, 2000.
17. T.F. Cootes, G.H. Edwards, and C.J. Taylor, "Active appearance models," in *Proceedings of the European conference on Computer Vision*, 1998, vol. 2, pp. 484–498.
18. Arnaud Doucet, Nando de Freitas, Kevin Murphy, and Stuart Russell, "Rao-blackwellised particle filtering for dynamic bayesian networks," in *Proc. of Uncertainty in AI*, 2000.
19. Kevin Murphy and Stuart Russell, *Sequential Monte Carlo Methods in Practice*, chapter Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks.
20. J. Besag, "On the statistical analysis of dirty pictures," *Journal of the Royal Stat. Soc.*, vol. 48, no. 3, pp. 259–302, 1986.