Learning Stochastic Path Planning Models from Video Images

Technical Report UCI-ICS 04-12
School of Information and Computer Science
University of California, Irvine

Sridevi Parise, Padhraic Smyth
Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425
{sparise,smyth}@ics.uci.edu

June 18, 2004

Abstract

We describe a probabilistic framework for learning models of pedestrian trajectories in general outdoor scenes. Possible applications include simulation of motion in computer graphics, video surveillance, and architectural design and analysis. The models are based on a combination of Kalman filters and stochastic path-planning via landmarks, where the landmarks are learned from the data. A dynamic Bayesian network (DBN) framework is used to represent the model as a position-dependent switching state space model. We illustrate how such models can be learned and used for prediction using the block Gibbs sampler with forward-backward recursions. The ideas are illustrated using a real world data set collected in an unconstrained outdoor scene.

1 Introduction

In this paper we describe a probabilistic landmark-based framework for modeling, learning, and predicting pedestrian trajectories in a scene (as obtained from video-based tracking, for example). Figure 1 shows an example of a set of trajectories. Trajectory modeling and prediction has applications in problems such as computer graphics, video surveillance, path usage analysis for planning and design, trajectory compression for logging purposes, and as an aid to handle occlusions or other ambiguities during tracking.

In prior work on pedestrian trajectory modeling clustering algorithms have been used to model different behaviors (e.g., [Walter et al., 1999, Makris and Ellis, 2002, Bennewitz et al., 2002]). Our work differs from the clustering approaches in that we explicitly model goal-directed behavior via the notion of "landmarks" (or intermediate goals), imparting a richer representational structure to the model. In [Patterson et al., 2003] models of pedestrian motion trajectories are learned from GPS sensor data. Unlike our work, there is no notion of a final goal, and landmarks are fixed a priori based on already existing street maps. A similar model is proposed in [Bui et al., 2001], using a discretized space representation and an assumption that the trajectories are already segmented in terms of landmark boundaries for learning purposes. In contrast to this earlier work we learn a model (including landmark locations) in an unsupervised manner. In addition, rather than using a fixed set of pre-specified landmarks that are same for all trajectories, we allow a probabilistic hierarchical structure for each landmark such that the precise locations of the intermediate landmarks are allowed to vary by individual

The rest of the paper is organized as follows. In section 2 we describe the general model. Sections 3 and 4 discuss inference and learning in this type of model using Gibbs sampling and the Monte Carlo EM algorithm. Section 5 describes experiments and results and section 6 concludes the paper with a discussion on possible future work.

2 The Model

2.1 Assumptions

We make the following assumptions about pedestrian motion.

- The scene has a fixed set of *end regions* from which pedestrians can enter or exit (e.g., walkways on the edge of the scene, doors to buildings, etc).
- The scene has a fixed set of *landmark regions.* These represent regions where pedestrians can change direction and act as intermediate goals, playing an important part in their path planning.
- People move in approximately straight lines with constant speed towards target landmark regions.
- Humans exhibit a goal directed behavior.

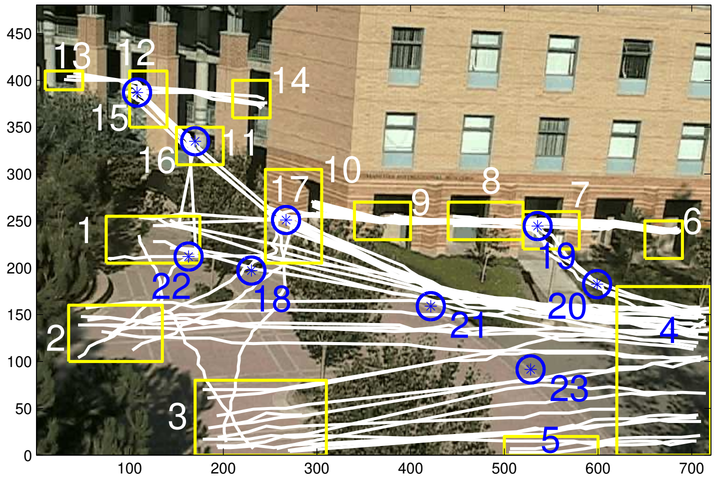


Figure 1: Example scene: The white lines represent pedestrian trajectories. The rectangles represent entry/exit regions and the *'s and circles represent initial guesses (from a heuristic non-probabilistic algorithm) for the means and covariances of the landmark regions. The entry/exit regions are numbered from 1 to 14 and the landmark regions are numbered from 15-23

These assumptions are clearly a gross simplification of realistic pedestrian motion (e.g., there is no concept of interactions among pedestrians, speed can vary), but nonetheless are intended as a first-step in building models that can approximate the general characteristics of pedestrian trajectories and that can be learned from data.

2.2 Trajectory Generation

Given the above assumptions, the model generates trajectories as follows:

- At time $t = 0$, choose a start region $s0$ from the set of entry/exit regions according to a probability distribution $\pi(s0)$. $\pi(i)$ is a multinomial distribution that gives the probability of starting in any end region i.
- Choose an exit/goal region sE according to $E(s0, sE)$. $E(s0, sE)$ is a probability transition matrix, such that,

$$E(i, j) = p(exit \ = \ j | start\ region \ = \ i)$$

- Choose a speed v from the population distribution on speeds. Here we assume a truncated Gaussian, $\mathcal{N}(\mu_v, \sigma_v^2)$, truncated to disallow negative speeds.
- At each time step t, we assume the existence of an active landmark region L_t (explained below), and we take a noisy step with speed v toward a target point $\boldsymbol{l_{L_t}}$ (explained below) associated with this landmark region to give the position, $\boldsymbol{b_t}$. Here, $\boldsymbol{l_{L_t}}$ and $\boldsymbol{b_t}$ are two-dimensional vectors representing (x, y) positions in the image. We assume additive zero

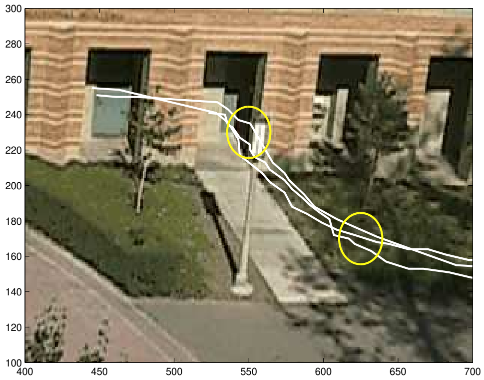


Figure 2: An illustration of the need for different target points within landmark regions

mean Gaussian noise.

$$p(\boldsymbol{b_t}|\boldsymbol{b_{t-1}}, v, L_t) = \mathcal{N}(\boldsymbol{b_{t-1}} + \begin{bmatrix} cos\theta_{\boldsymbol{b_{t-1}},L_t} \\ sin\theta_{\boldsymbol{b_{t-1}},L_t} \end{bmatrix} v, \Sigma_b),$$

where, $\theta_{\boldsymbol{b_{t-1}},L_t}$ is the slope of the line, $(\boldsymbol{l_{L_t}} - \boldsymbol{b_{t-1}})$ and Σ_b is the covariance of the Gaussian noise.

- In this model, L_t is a multinomial random variable representing the landmark region towards which the individual is moving during time t. We assume that each landmark region is represented by a target point when an individual is moving towards that landmark region. The individual selects the next active landmark region at a given time step t, $L_{(t+1)}$, depending on the current position, the current landmark region (including its target point) and the final goal region. More specifically, we introduce a switch variable T_{t+1} that takes values from $\{0, 1\}$, indicating the presence or absence of a landmark region switch (or turn) and define a probability of switching landmark regions as a function of the distance between the current position and current landmark region target point,

$$p(T_{t+1} = 1|L_t, b_t) = \frac{1}{1 + exp(||\boldsymbol{l_{L_t}} - \boldsymbol{b_t}|| - \beta)},$$

where β is some parameter controlling the range of the turn. If this distance is small enough to trigger a switch, we choose the next active landmark region (which could be an end region) using a 2nd order transition probability matrix conditioned on both the current landmark region and the goal state (the end region). If there is no switch, we keep the current landmark region active. That is,

$$\begin{aligned} p(L_t = i|L_{t-1} = j, T_t, S_E) &= trans(j, i, S_E), \quad \text{if } \; T_t = 1 \\ &= \delta(i, j), \quad \text{if } \; T_t = 0 \end{aligned}$$

where $trans$ is the 2nd order transition probability matrix, so that,

$$\sum_i trans(j, i, S_E) = 1$$

In the following text, we sometimes use the term landmark region to refer to both landmark regions and end regions - the usage should be clear from context.

- We observe noisy measurements, y_t, of the actual positions, again with additive zero mean Gaussian noise.

$$p(y_t|b_t) = \mathcal{N}(b_t, \Sigma_e),$$

  where Σ_e is the noise covariance matrix.

For the landmark regions, instead of fixing these to be single points with the same target point for all the trajectories, we allow these to be defined by extended regions, each trajectory picking a set of individual target points for use from these regions. This hierarchical structure on the landmarks seems to capture the natural structure in the data as seen in figure 2. Here we can see that different trajectories have different target points for a given landmark region (indicated by circular regions). In particular, we represent the end regions as rectangles with uniform probability of picking a target point. The distribution of possible targets points for each internal landmark region is represented as a 2-dimensional Gaussian with a mean location and covariance matrix.

2.3 Model Parameters

The various parameters mentioned in the previous section, that need to be specified in order to generate a trajectory from the model, are summarized below.

1. $\pi(i)$, which defines the probability of starting in any end region i.
2. $E(i,j) = p(goalregion = j|startregion = i)$
3. $trans(i,j,e)$, which defines the probability of the next landmark region when there is a landmark region switch, given the current landmark and the goal.
4. μ_v and σ_v, the parameters of the population speed.
5. Σ_b, the covariance matrix for the Gaussian errors in the dynamics
6. Σ_e, the covariance matrix for the Gaussian errors in the measurements.
7. β, the parameter controlling the landmark region switching range.
8. The parameters giving the four corners of each rectangle defining the end regions.
9. $\{\mu_l^{(i)}, \Sigma_l^{(i)}\}$, the mean and covariances defining the Gaussian distributions for the target points of a landmark region i.

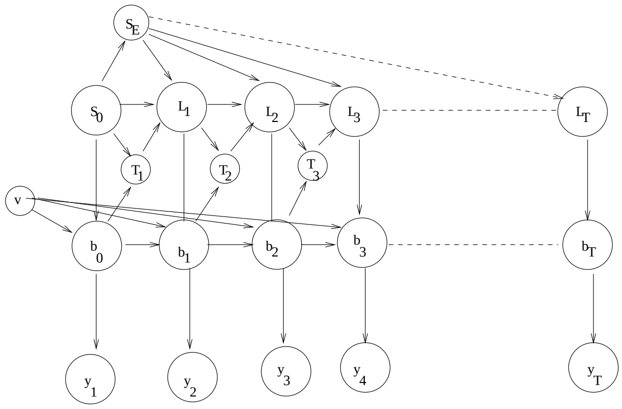


Figure 3: DBN for our model

2.4 Representation as a Dynamic Bayesian Network

To aid in understanding the various dependency relations between variables in the model, it is useful to represent it as a dynamic Bayesian network (DBN) as shown in Figure 3. Here the variables at each time slice t are $\{L_t, T_t, b_t, y_t\}$. We also have v and s_E that influence the entire trajectory. In this figure, for clarity as well as to emphasize the basic intermediate landmark region switching structure, we have not included the random variables that represent the locations or target points of the landmark regions and entry/exit regions for an individual trajectory—there is a population prior for each, but for any individual trajectory the exact locations are not known. This graphical model implicitly assumes that the landmark regions are defined by single known points that are same for all trajectories. See figure 4 for the DBN that encodes the hierarchical landmark structure.

3 Inference given Known Parameters

Our model is similar to a switching state-space model [Ghahramani and Hinton, 1996], with the additional dependence of switch variables on current position. Exact inference in this model is not tractable because of the presence of the switching variables (See Appendix for details). Hence, we resort to approximate inference techniques.

3.1 Offline Inference and Forward-Backward Gibbs Sampling

For offline inference, we use a block Gibbs sampler based on the forward filtering and backward sampling recursions of [Carter and Kohn, 1994]. We call this the forward-backward (FB) Gibbs sampler, following [Scott, 2002]. Referring back to figure 3, we would like to infer all the hidden sates given an entire observation sequence, that is, we want to generate samples from $p(L_{1:T}, T_{1:T}, b_{0:T-1}, v|S_0, S_E, y_{0:t})$ (here we assume we know the start and end regions). We do this by sampling, $\{L_{1:T}, T_{1:T}\}, \{b_{0:T}\}$ and $\{v\}$ as separate blocks from the appropriate conditional distributions. That is, we cycle through the following steps:

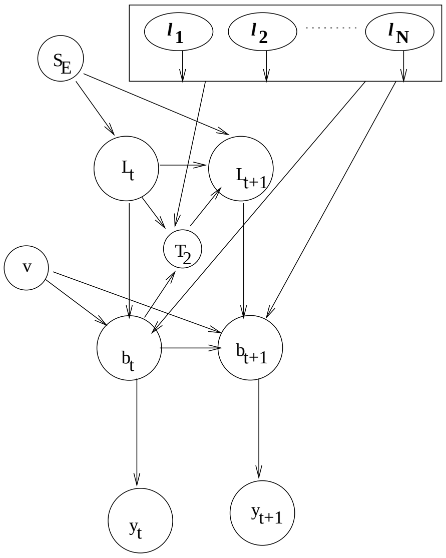


Figure 4: 2-slice DBN for model with the hierarchical landmark structure: $\boldsymbol{l_i}$ denotes the target point for landmark region i for this trajectory. N denotes the total number of landmark regions.

1. generate samples from $p(\{L_{1:T}, T_{1:t}\}|\text{all other variables})$
2. generate samples from $p(b_{0:T}|\text{all other variables})$
3. generate samples from $p(v|\text{all other variables})$

Note that, it is important to block the variables in this way, since in our model the samples for variables at neighboring time steps will be highly correlated and sampling them individually would make it difficult for the sampler to move away from any given configuration. Also, a good initialization is important and we initialize the b chain at the observations and v to be the average speed as calculated from $y_{1:T}$. We assume we know S_0 and S_E which can be reasonably assigned based on y_0 and y_T.

The various sampling steps involved are further explained below.

3.1.1 Sampling the $\{L, T\}$ chain

We can generate samples from $p(\{L_{1:T}, T_{1:t}\}|\text{all other variables})$ as follows:

Recursive Forward Filtering Pass: to compute $p(L_t, T_t|y_{0:t}, b_{0:t}, S_0, S_E, v)$, for $t = 1, 2, \ldots, T$.

$$\begin{aligned} p(L_t, T_t|y_{0:t}, b_{0:t}, S_0, S_E, v) &= \sum_{L_{t-1}} p(L_t, T_t, L_{t-1}|y_{0:t}, b_{0:t}, S_0, S_E, v) \\ &\propto p(b_t|L_t, b_{t-1}, v)X \\ &\quad X \sum_{L_{t-1}} p(L_{t-1}|y_{0:t-1}, b_{0:t-1}, S_0, S_E, v)p(T_t|L_{t-1}, b_{t-1})X \\ &\qquad\qquad X p(L_t|T_t, L_{t-1}, S_E) \end{aligned}$$

In the above, $p(L_{t-1}|y_{0:t-1}, b_{0:t-1}, S_0, S_E, v)$ can be obtained by summing out T_{t-1} in $p(L_{t-1}, T_{t-1}|y_{0:t-1}, b_{0:t-1}, S_0, S_E, v)$ which we already have from the forward computation at time $t-1$.

Backward pass: We successively generate samples from the required joint in a backward pass by factorizing the joint as follows:

$$\begin{aligned} p(L_{1:T}, T_{1:T}|b_{0:T}, y_{0:T}, v, S_0, S_E) &= p(L_T, T_T|b_{0:T}, y_{0:T}, v, S_0, S_E)X \\ &\quad X \prod_{t=T-1}^{1} p(L_t, T_t|b_{0:T}, y_{0:T}, v, S_0, S_E, L_{t+1:T}, T_{t+1:T}) \end{aligned}$$

We already have, $p(L_T, T_T|b_{0:T}, y_{0:T}, v, S_0, S_E)$ from the forward pass, and also,

$$p(L_t, T_t|b_{0:T}, y_{0:T}, v, S_0, S_E, L_{t+1:T}, T_{t+1:T}) \propto p(L_{t+1}, T_{t+1}|L_t, b_t, S_E)p(L_t, T_t|b_{0:t}, y_{0:t}, v, S_0, S_E)$$

3.1.2 Sampling v

Since b_{t+1} depends linearly on v with Gaussian noise and also since we assume that the prior distribution on v is Gaussian, we can recursively update the posterior distribution, $p(v|\text{all other variables})$, in closed form.

$$p(v|\text{all other variables}) = p(v|b_{0:T}, L_{0:T})$$

Now,

$$\begin{aligned} p(v|b_{0:t}, L_{0:t}) &\propto p(b_t, L_t|b_{0:t-1}, L_{0:t-1}, v)p(v|b_{0:t-1}, L_{0:t-1}) \\ &\propto p(b_t|L_t, b_{t-1}, v)p(v|b_{0:t-1}, L_{0:t-1}) \end{aligned}$$

Let,

$$p(v|b_{0:t-1}, L_{0:t-1}) \sim \mathcal{N}(\mu_v^{(t-1)}, \sigma_v^{2(t-1)})\text{and}$$

$$p(v|b_{0:t}, L_{0:t}) \sim \mathcal{N}(\mu_v^{(t)}, \sigma_v^{2(t)})$$

We have,

$$b_{t+1} = C_{L_{t+1},b_t} v + b_t + e_b, \text{ where } e_b \sim \mathcal{N}(0, \Sigma_b)$$

Here, $C_{L_{t+1},b_t} = \begin{bmatrix} cos\theta \\ sin\theta \end{bmatrix}_{L_{t+1},b_t}$ gives the direction towards the currently active landmark region.

Following on the lines of the Kalman update equations, we have,

$$\begin{aligned} \mu_v^{(t)} &= \mu_v^{(t-1)} + K_t(b_t - \{C_{L_t,b_{t-1}} v + b_{t-1}\}) \\ \sigma_v^{2(t)} &= (1 - K_t C)\sigma_v^{2(t-1)} \\ K_t &= \sigma_v^{2(t-1)} C^T (C\sigma_v^{2(t-1)} C^T + \Sigma_b)^{-1} \end{aligned}$$

3.1.3 Sampling the b-chain

When generating samples for the b-chain from $p(b_{0:T}|\text{all other variables})$, we cannot directly apply the FB algorithm, since the forward distributions required for the b's are not available in closed form due to the position-dependent switching. We overcome this problem by maintaining and propagating a sample based approximation to this forward distribution as explained below.

Recursive Forward Filtering Pass: to compute $p(b_t|y_{0:t}, L_{0:t}, T_{1:t+1}, S_E, v)$, for $t = 1, 2, \ldots, T$.

$$\begin{aligned} p(b_t|y_{0:t}, L_{0:t}, T_{1:t+1}, S_E, v) \quad & \propto p(y_t|b_t) p(T_{t+1}|L_t, b_t) \\ \times \int_{b_{t-1}} p(L_t|L_{t-1}, T_t) \quad & p(b_{t-1}|L_{0:t-1}, T_{1:t}, y_{0:t-1}, S_E, v) p(b_t|b_{t-1}, L_t, v) \end{aligned}$$

Given the above equation, we use importance sampling to recursively generate a weighted sample set that approximates the above filtered distribution at time t using samples from the corresponding distribution at time $(t-1)$.

Backward Pass:

$$\begin{aligned} p(b_{0:T}|S_0, S_E, L_{1:T}, T_{1:T}, v, y_{0:T}) = \quad & p(b_T|S_0, S_E, L_{1:T}, T_{1:T}, v, y_{0:T}) \\ \times \quad & \prod_{t=T-1}^{0} p(b_t|b_{t+1:T}, S_0, S_E, L_{1:T}, T_{1:T}, v, y_{0:T}), \quad \text{and} \end{aligned}$$

$$\begin{aligned} p(b_t|b_{t+1:T}, S_0, S_E, L_{1:T}, T_{1:T}, v, y_{0:T}) &= p(b_t|b_{t+1}, S_0, S_E, L_{1:t+1}, T_{1:t+1}, v, y_{0:t}) \\ &\propto p(b_{t+1}|L_{t+1}, b_t, v) p(b_t|S_0, S_E, L_{1:t}, T_{1:t+1}, v, y_{0:t}) \end{aligned}$$

Given a sample for b_{t+1}, we update the filtered distribution for b_t according to the above equations (by updating the weights appropriately) and then draw a sample from it.

3.1.4 Sampling the landmark region target points

With the hierarchical structure on landmark regions, we have the target points as additional random variables in the model as seen in figure 4. When these are unknown, they can be inferred by sampling them as additional blocks in the gibbs sampling. We assume that we know the target points for the end regions. Note that only one end region target point is required - that associated with the known goal region S_E. We assign it to be y_T and denote it explicitly by $target\_sE$. For target points of internal landmark regions, we can sample each from the required conditional, $p(\boldsymbol{l_i}|\text{all other variables})$, as follows:

Recursive Forward Filtering Pass: to compute $p(\boldsymbol{l_i}|y_{0:t}, L_{0:t}, T_{1:t}, S_E, v, b_{0:t}, target\_sE, \{\boldsymbol{l_{j\setminus i}}\})$, for $t = 1, 2, \ldots, T$. Here $\{\boldsymbol{l_{j\setminus i}}\}$, denotes all internal landmark region target points except $\boldsymbol{l_i}$.

$$
\begin{aligned}
p(\boldsymbol{l_i}&|y_{0:t}, L_{0:t}, T_{1:t}, S_E, v, b_{0:t}, target\_sE, \{\boldsymbol{l_{j\setminus i}}\}) \\
&\propto p(L_t, T_t, b_t, y_t|L_{0:t-1}, T_{1:t-1}, v, y_{0:t-1}, S_E, b_{0:t-1}, target\_sE, \{\boldsymbol{l_j}\})\times \\
&\qquad \times p(\boldsymbol{l_i}|L_{0:t-1}, T_{1:t-1}, v, b_{0:t-1}, y_{0:t-1}, S_E, target\_sE, \{\boldsymbol{l_{j\setminus i}}\}) \\
&\propto p(T_t|L_{t-1}, b_{t-1}, \{\boldsymbol{l_j}\}).p(b_t|L_t, b_{t-1}, v, \{\boldsymbol{l_j}\}) \times \\
&\qquad p(\boldsymbol{l_i}|L_{0:t-1}, T_{1:t-1}, v, b_{0:t-1}, y_{0:t-1}, S_E, target\_sE, \{\boldsymbol{l_{j\setminus i}}\})
\end{aligned}
$$

Since the above distribution cannot be obtained in closed form, we maintain a sampled representation as we do for the b-chain.

The backward pass is also similar to that for the b-chain. Also note that sampling of each landmark region position does not effect the conditional distributions of other landmark region positions when these are sampled consecutively. Hence, we can update the required distributions for all landmark region positions simultaneously by going through each trajectory just once and then sampling all these positions together.

3.2 Online Inference and Particle Filters

Particle filtering is an alternate sampling method [Doucet et al., 2001]. Although the basic particle filter (or bootstrap filter) [Doucet et al., 2001] has broad applicability and in theory can be applied to any type of model, many adjustments and refinements may be required for good performance in practice. In our model, the observations have a direct impact only on the positions, b_t, and not on the landmark sequence. In the absence of a good proposal distribution, this leads to inaccurate representations of the posterior distributions for variables other than b_t. Also the presence of static variables such as the goal state and speed and also slowly varying parameters such as active landmark variables effect the performance of the particle filter since the support for these does not change once they are sampled. These factors also necessitate the need for a careful choice of resampling scheme inorder to prevent premature elimination of good particle paths in cases where the initial trajectory is in low probability regions. The effect of the above problems is especially evident in the case where we do not already have the model parameters and are trying to learn their values from data, by embedding the particle filter in an intermediate inference step. Thus, for learning, we found the FB Gibbs sampling to be more robust in general.

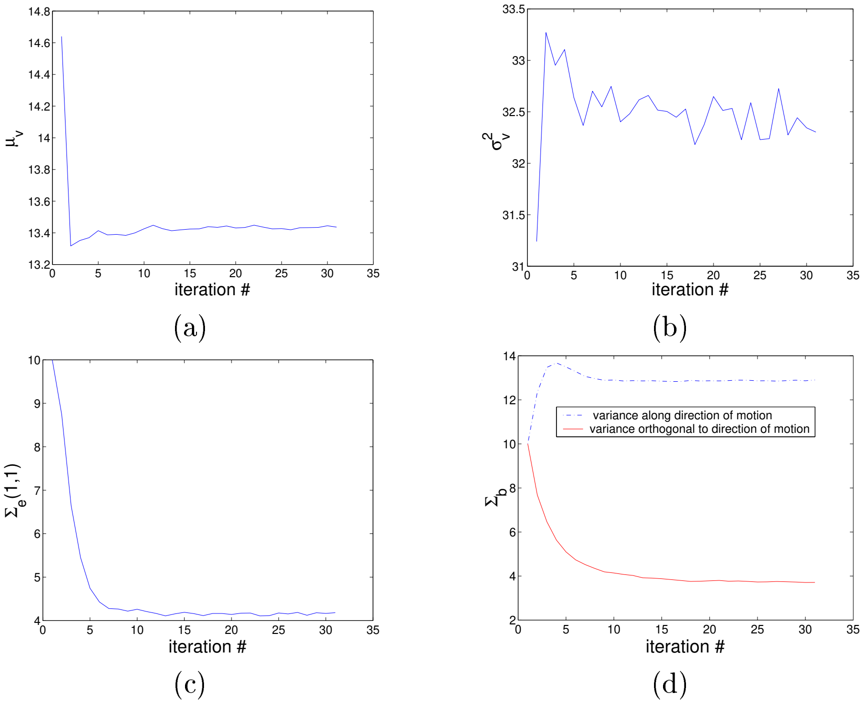


Figure 5: Learning with MCEM: (a) and (b) show the variation of mean and variance respectively of the population speed with iteration number. (c) shows the variation of the parameter Σ_e with iteration number. (d) shows the variation of the parameter Σ_b with iteration number.

However, a major advantage of particle filters over Gibbs sampling is that the posterior is updated recursively over time. This makes them very efficient for online inferencing tasks where we need to compute this distribution at each time step. So for online inferencing tasks where we assume we already learned a good set of parameters, we use the particle filter. To counter some of the above mentioned problems, we use the balanced resampling scheme suggested in [Liu et al., 2001]. Here, unlike the basic multinomial resampling, the relationship between weights and the number of resampled copies need not be linear and can be adjusted (the parameter α) to balance the need for diversity and focus. For our experiments, we use ($\alpha = 0.1$). To handle the static variables, we use the static parameter rejuvenation scheme suggested in [Storvik, 2002]. This allows for sampling the static variables at each time step independent of their values at the previous time steps based on the values of the other variables in the model. For computational efficiency [Storvik, 2002] requires the presence of sufficient statistics that can be recursively updated. It is easy to see that this is true for our static variables v and sE. For v, the recursive computation here is similar to its forward computation in the gibbs sampling and for sE the sufficient statistic is the different landmark region switchings and this information can be updated recursively at each t. To further improve the performance, we interleave the particle filter updates with occasional FB Gibbs sampling runs at intermediate time steps. We use a deterministic schedule, though one can think of using some heuristic based dynamic schedule. These FB runs help the particle filter to refocus in the right regions in cases where it has drifted.

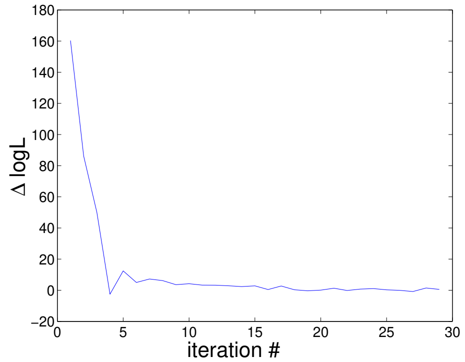


Figure 6: Learning with MCEM: variation of the estimate for the increase in log likelihood obtained by bridge sampling with iteration number. It converges to zero from above with some random variation, indicating approximate convergence.

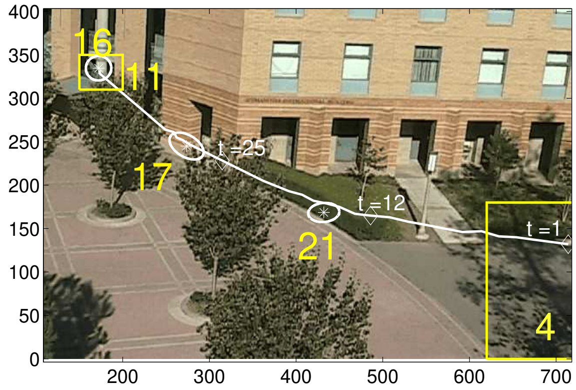
4 Learning the Model Parameters

Given a set of video sequences showing pedestrian motion in a scene, we estimate the model parameters using a Monte Carlo version of the EM (MCEM) algorithm [Wei and Tanner, 1990]. We use the afore-mentioned FB Gibbs sampling scheme to approximate the E step.

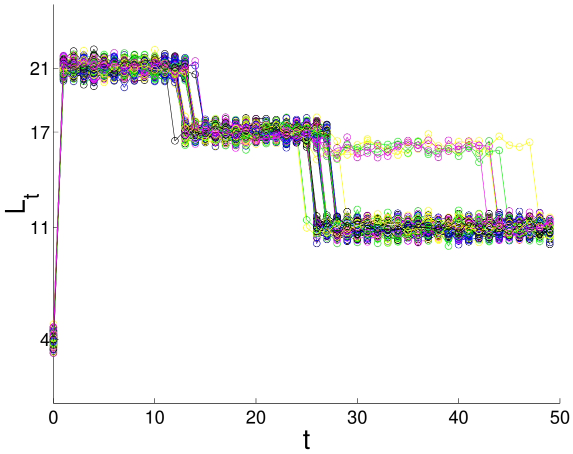
The data used for learning consists of a set of 67 manually tracked trajectories for the example scene shown in figure 1 using video data from single fixed-location camera. For each frame the estimated x,y location of a pedestrian's head in the image was located by manual mouse-clicks. Gaps in the trajectories due to occlusions were filled in with equidistant observations on interpolated straight lines.

For this scene, we selected the number of end regions (14) and the number of landmark regions (9) subjectively. With sufficient computational power one could also in principle learn the numbers of each (e.g., find the number of landmark regions that give the best predictive power on out-of-sample data). The end regions are assumed to be specified by the user and are not altered during learning. These are defined as rectangular regions and are mostly at the scene boundaries and other possible entrances such as doors etc. in the scene. The initial estimates for the locations of the means for various landmark regions are obtained by running a simple linear segmentation algorithm [Mann et al., 2002] on each trajectory to find the linear segments and corresponding switch points and then clustering those points where segments have considerable change in direction (we use $> 10°$). [Mann et al., 2002] uses a parameter λ that defines the trade-off between linearity and number of segments. We use $\lambda = 500$. The initial estimates of locations of various end regions and landmark regions are shown in figure 1.

Given this approximate (non-probabilistic) segmentation of trajectories, one can generate initial maximum likelihood estimates for the path planning parameters. We smooth these using some heuristics, such as preferring the shortest straight line path to the goal, assigning a non-zero probability of using a previously unused segment in one direction given that it has been used in the other direction etc., to use as an initialization for the MCEM algorithm.



(a)

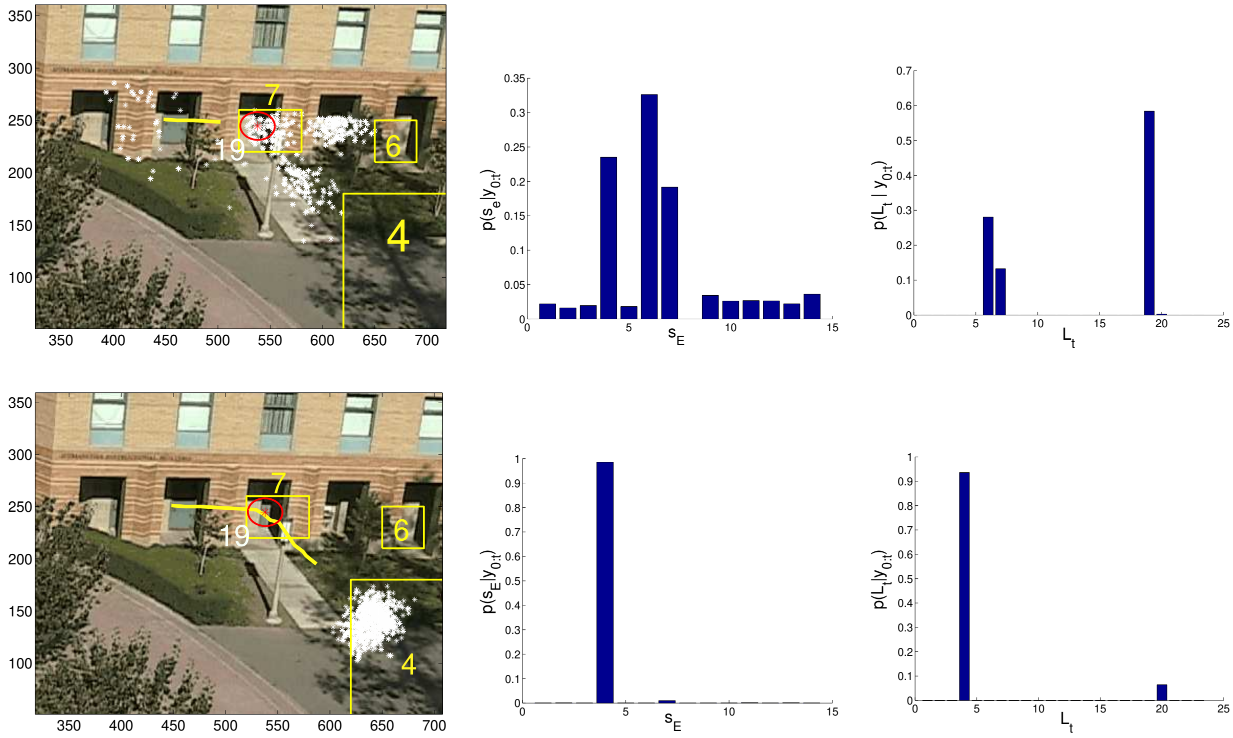


(b)

Figure 7: Trajectory segmentation using FB: (a) shows the trajectory and relevant landmark regions and some intermediate time points. (b) shows the samples for the L-chain obtained from the Gibbs sampler (slightly jittered to show all samples). From this we can see that the most likely segmentation for the trajectory is [4 21 17 11] (Note that there is some probability of going from 17 to 16. This is because 16 and 11 (an end region) overlap).

For the noise in the dynamics, Σ_b, we use a covariance matrix that is described by two parameters— the variance along the direction of motion and variance orthogonal to the direction of motion, and diagonal along the direction of motion. Hence, in the original coordinate frame we get a varying Σ_b with t. The measurement noise covariance Σ_e is assumed to be isotropic.

We estimate all the parameters described in section 2.3, except the end region definitions and the parameter β, which we treat as fixed. It was seen empirically that $\beta = 40$ pixels gives a good segmentation. Obtaining initial parameter values for other variables as described above (for parameters not mentioned above, such as for example, the covariance of the landmark regions, we use a reasonable ad hoc guess), we ran the MCEM algorithm on the trajectory data. For the gibbs sampling we used a sample size of 100 and for the sampled representation of the forward distributions we again used a sample size of 100. The convergence of the MCEM algorithm is monitored using bridge sampling [Meng and Schilling, 1996]. We also assess convergence by looking at the values of the various parameters at each iteration. We could see that the parameters of the speed and noise covariances start converging after just a few iterations (around 15). See figure 5. However, the landmark region distributions did not seem to converge even after 30 iterations. This issue needs further investigation. Currently, we avoid this problem by learning the landmark region distributions for first few iterations (15) and then clamping their values. Figure 6 shows the estimate for increase in loglikelihood at each iteration as given by bridge sampling. We can see that this is a curve converging to zero from above with some random variation. This shows that approximate convergence has been reached.



(a) (b) (c)

Figure 8: Online Trajectory Monitoring: Each row corresponds to a given time t. (a) shows the partial trajectory up to t and the samples for the 7-step ahead predictions. Also the relevant landmark regions are marked. (b) and (c) show the probability of the goal state and the current active landmark region given observations up to t respectively.

5 Experiments

In this section, we show the learned models from section 4 can be useful for various applications.

We can use FB Gibbs sampling to infer the underlying L-chain given an entire observation sequence and the learned parameters. See figure 7 for an example. This kind of landmark-based representation of the trajectories is useful in many applications, for example, for trajectory data compression as suggested in [Makris and Ellis, 2002], for path based clustering or for analyzing path usage. In some environments, each segment may have a natural interpretation as being associated with a specific behavior or activity.

The model can also provide useful information about a trajectory's future behavior given the partial trajectory up to current time. It can provide high level information such as the distribution over the trajectory's possible exit region and currently active landmark region. These could be useful in online activity prediction tasks. Also long range predictions made by the model can aid any tracking process, especially in the presence of occlusions and turns.

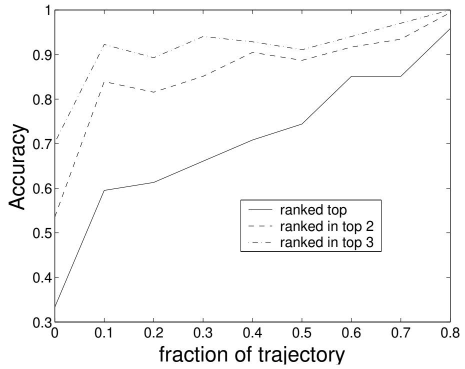


Figure 9: Accuracy in predicting the end region as a function of the fraction of measurements available from the full trajectory.

As mentioned in section 3, we use the particle filter (with static parameter rejuvenation and occasional FB runs) for these online inference tasks. The number of particles used was 4000. At each time step t, the particle filter gives a sample(weighted) based representation of the filtered distribution on all the hidden states. For the multi-step ahead prediction, at each t, we extend the particle paths from the filtered distributions by just using the system dynamics without any observations. These ideas are illustrated in figure 8. From the figure, we can see that for the multi-step ahead predictions, the model is able to provide multi-modal distributions to account for possible turns or end regions along the path. This would not be possible with a simple model that has no notion of landmarks and landmark switching.

Figure 9 shows the accuracy of predicting the true goal state at specific times, averaged over 28 test trajectories (these were not included in training and recorded several months after the original training data was collected). Since the trajectories are of different lengths, we use times defined by the fraction of the trajectory's length rather than absolute values. To predict the goal at any time t, we sort the end regions e according to $p(s_E = e|y_{0:t})$ and pick the top i regions as the prediction set for making the "top i" prediction. The default accuracy at time 0 based on knowing only where the pedestrian entered the scene was about 0.33.

6 Conclusions

In this paper, we described a model for pedestrian trajectories and discussed learning of such a model from data. This work represents an early step in the direction of more sophisticated models for population motion that can be learned automatically from data. Despite the simplicity of the model, we are encouraged by the fact that the models appear to capture well the gross characteristics of motion in an outdoor scene and that in principle all aspects of the model can be learned in an unsupervised manner. Directions for future work involves developing more accurate richer models including mechanisms to handle multiple individuals, interactions, "stop and go" behaviors, and appropriate methods for occlusion. We also plan to look at ways to automatically chose the number of landmark regions during learning.

Acknowledgments

This work was supported by the National Science Foundation under grant number UCI-0331707. We also gratefully acknowledge the assistance of Momo Alhazzazi in obtaining and preprocessing of the video data.

A Need For Approximate Inference

Suppose that, given the model in figure 3, we would like to do inference. The most common inference tasks in these kind of state space models involve finding the filtered and smoothed distributions of the hidden states given a set of observations. That is, finding $p(h_t|y_{0:t})$ and $p(h_t|y_{0:T})$, where h_t denotes the hidden state at time t. (These are also required for the E step in an EM algorithm for learning the model parameters). Usually these are computed by carrying out a set of forward-backward recursions (for example, such as the $\alpha\beta$ procedure for HMMs).

Let us group the variables, so that, $h_t = \{L_t, T_{t+1}, b_t\}$. Consider the recursive forward computation of the quantity, $p(h_t|y_{0:t})$:

$$\begin{aligned} p(h_t|y_{0:t}) &= \int_{h_{t-1}} p(h_t, h_{t-1}|y_{0:t}) \\ &= \int_{h_{t-1}} \frac{p(y_t|h_t)p(h_{t-1}|y_{0:t-1})p(h_t|h_{t-1})}{p(y_t|y_{0:t-1})} \\ &= \frac{p(y_t|b_t)}{p(y_t|y_{0:t-1})} \sum_{L_{t-1}} \sum_{T_t} \int_{b_{t-1}} p(L_{t-1}, T_t, b_{t-1}|y_{0:t-1}) p(L_t|L_{t-1}, T_t, S_E) p(b_t|L_t, b_{t-1}) \times \\ &\qquad \times p(T_{t+1}|L_t, b_t) \end{aligned} \tag{1}$$

Say, you are given that,

$$\begin{aligned} p(L_0, T_1, b_0|y_0) &= \mathcal{N}(b_0|\mu_0, \Sigma_0), \text{ if } L_0 = S_0 \text{ and } T_1 = 1 \\ &= 0, \text{otherwise} \end{aligned}$$

(In the above, μ_0 could be a function of y_0 and S_0)

From (1) for ($t = 1$), $p(L_1, T_2, b_1|y_0, y_1)$ for fixed values of the discrete variables L_1, T_2 is given by:

$$\begin{aligned} p(L_1, T_2, b_1|y_0, y_1) &= \frac{p(y_1|b_1)}{p(y_1|y_0)} \sum_{L_0} \sum_{T_1} p(L_1|L_0, T_1, S_E) \int_{b_0} p(L_0, T_1, b_0|y_0) p(b_1|L_1, b_0) \times \\ &\qquad \times p(T_2|L_1, b_1) \qquad (2) \\ &\propto p(y_1|b_1) p(L_1|L_0 = S_0, T_1 = 1, S_E) \left[\int_{b_0} \mathcal{N}(b_0|\mu_0, \sigma_0) p(b_1|L_1, b_0) \right] p(T_2|L_1, b_1) \\ &\propto \mathcal{N}(b_1|\mu_1, \Sigma_1) p(T_2|L_1, b_1), \text{ where } \mu_1 \text{ depends on } L_1, \mu_0 \text{ and } y_1 \\ &\propto exp(-\frac{1}{2}(b_1 - \mu_1)^T \Sigma_1^{-1} (b_1 - \mu_1) \times exp(-\log(1 + exp(||L_1 - b_1|| - \beta))), \\ &\qquad , \text{if } T_2 = 1 \end{aligned}$$

Since we cannot evaluate the above distribution in closed form, we need some kind of approximation. Also, though not obvious for $t = 1$ since L_0, T_1 can take only one value, for other $t > 1$, we can see that the sums in (2) increase the number of components for the distribution of b_t by an order of L times, for each t, where L is the number of landmark regions. This exponential growth in the complexity of the state with time is a known problem, that makes exact inference intractable, in switching state space models [Ghahramani and Hinton, 1996]. To overcome these problems with exact inferencing in our model, we use the sampling approach described in the paper.

References

M. Bennewitz, W. Burgard, and S. Thrun. Using EM to learn motion behaviors of persons with mobile robots. In *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, 2002.

H. H. Bui, S. Venkatesh, and G. West. Tracking and surveillance in wide-area spatial environments using the abstract hidden markov model. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1):177–195, 2001.

C. K. Carter and R. Kohn. On Gibbs sampling for state space models. *Biometrika*, 81(3):541–553, 1994.

A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice.* Springer, New York, 2001.

Z. Ghahramani and G. E. Hinton. Switching state space models. *Technical Report CRG-TR-96-3, Dept. Comp. Sci., University of Toronto*, 1996.

J. S. Liu, R. Chen, and T. Logvinenko. A theoretical framework for sequential importance sampling with resampling. In *Sequential Monte Carlo Methods in Practice*, pages 225–246. Springer, 2001.

D. Makris and T.J. Ellis. Spatial and probabilistic modeling of pedestrian behaviour. In *British Machine Vision Conference*, pages 557–566, 2002.

R. Mann, A. D. Jepson, and T. El-Maraghi. Trajectory segmentation using dynamic programming. In *International Conference on Pattern Recognition*, 2002.

X. Meng and S. Schilling. Fitting full-information item factor models and an emperical investigation of bridge sampling. *Journal of the American Statistical Association*, 91(435):1254–1267, September 1996.

D. J. Patterson, L. Liao, D. Fox, and H. Kautz. Inferring high-level behavior from low-level sensors. In *UbiComp 2003: Ubiquitous Computing: 5th International Conference*, pages 73–89, 2003.

S. L. Scott. Bayesian methods for hidden Markov models, recursive computing in the 21st century. *Journal of the American Statistical Association*, 97:337–351, 2002.

G. Storvik. Particle filters for state-space models with the presence of unknown static parameters. *IEEE Transactions on Signal Processing*, 50(2):281–289, february 2002.

M. Walter, A. Psarrou, and S. Gong. Learning prior and observation augmented density models for behaviour recognition. In *British Machine Vision Conference*, pages 23–32, 1999.

G. Wei and M. Tanner. A Monte Carlo implementation of the EM algorithm and the poor mans data augmentation algorithms. *Journal of the American Statistical Association*, 85:699–704, 1990.