Tracking Multiple Mouse Contours (without Too Many Samples)

Kristin Branson and Serge Belongie
Dept. of Computer Science and Engineering
UC San Diego
La Jolla, CA 92093-0114

Abstract

We present a particle filtering algorithm for robustly tracking the contours of multiple deformable objects through severe occlusions. Our algorithm combines a multiple blob tracker with a contour tracker in a manner that keeps the required number of samples small. This is a natural combination because both algorithms have complementary strengths. The multiple blob tracker uses a natural multitarget model and searches a smaller and simpler space. On the other hand, contour tracking gives more fine-tuned results and relies on cues that are available during severe occlusions. Our choice of combination of these two algorithms accentuates the advantages of each. We demonstrate good performance on challenging video of three identical mice that contains multiple instances of severe occlusion.

1. Introduction

We address the problem of tracking the contours of multiple identical mice from video of the side of their cage; see Figure 3 for example frames. Although existing tracking algorithms may work well from an overhead view of the cage, the majority of vivaria are set up in a way that prohibits this view. A solution to the side view tracking problem would be very useful for medical researchers wishing to automatically monitor the health and behavior of lab animals [3].

This problem is also interesting and uniquely difficult from a computer vision standpoint. Because mice are highly deformable 3D objects with unconstrained motion, an accurate contour model is necessarily complex. Because mouse motion is erratic, the distribution of the current mouse positions given their past trajectories has high variance. The biggest challenge to tracking mice from a side view is that the mice occlude one another severely and often. Tracking the mice independently would inevitably result in two trackers following the same mouse. Instead, we need a multitarget algorithm that tracks the mice in concert. As the number of parameters that must be simultaneously estimated increases linearly with K, the number of mice, the search space size increases exponentially with K [13]. Thus, using existing approaches to directly search the contour space for all mice at once is prohibitively expensive.

In addition, tracking individual mouse identities is difficult because the mice are indistinguishable. We cannot rely on object-specific identity models (e.g. [4, 9]) and must instead accurately track the mice *during* occlusions. This is challenging because mice have few if any trackable features, their behavior is erratic, and edges (particularly between two mice) are hard to detect. Other features of the mouse tracking problem that make it difficult are clutter (the cage bedding, scratches on the cage, and the mice's tails), inconsistent lighting throughout the cage, and moving reflections and shadows cast by the mice.

Our algorithm is of general interest to the tracking community because the challenges to successful mouse tracking are common to many real world tracking applications. While many video sequence testbeds are constructed to show off the novelty of an algorithm, our algorithm is constructed to address the challenges of a specific tracking problem. Thus, our feature extraction algorithm must be powerful, our objects' state representation must be detailed, and our algorithm must be able to search the complex parameter space with a limited number of samples.

We propose a solution that combines existing blob and contour tracking algorithms. However, just combining these algorithms in the obvious way does not effectively solve the difficulties discussed above. We propose a novel combination of these algorithms which accentuates the strengths of each individual algorithm. In addition, we capitalize on the independence assumptions of our model to perform most of the search independently for each mouse. This reduces the size and complexity of the search space exponentially, and allows our Monte Carlo sampling algorithm to search the complex state parameter space with a reasonable number of samples. Our algorithm works with a detailed representation of a mouse contour to achieve encouraging results on a video sequence of three mice exploring a cage.

The paper is organized as follows. In Section 2, we describe the algorithms we build off of: the Bayesian Multiple Blob (BraMBLe) tracker [7] and MacCormick et al.'s contour likelihood model [12]. In Section 3, we describe the model assumed for the blob and contour tracking problem. In Section 4, we describe our particle filtering algorithm for fitting contours given this model. In Section 6, we present specific details of our algorithm, and the results for a challenging video sequence.

2. Previous Work

Our algorithm builds off of the Bayesian Multiple Blob (BraMBLe) tracker [7] and MacCormick et al.'s contour tracker [12, 2]. Both approaches are based on particle filtering. In this section, we first introduce standard particle filtering (a.k.a. bootstrap filtering, sequential importance sampling) to introduce our notation; see [5] for a complete treatment. Then, we describe and compare the blob and contour tracking algorithms.

2.1. Particle Filtering

Particle filtering is a sequential importance sampling algorithm for estimating properties of hidden variables given observations in a hidden Markov model. For tracking from video, $\mathbf{x}_t$, the state at time t, represents the positions of the objects in frame t, and $\mathbf{y}_t$, the observation at time t, is a function of video frame t. Standard particle filtering assumes that we can directly sample from $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, the density of transitioning to state $\mathbf{x}_t$ from $\mathbf{x}_{t-1}$. It also assumes that we can easily evaluate the observation likelihood, $p(\mathbf{y}_t|\mathbf{x}_t)$, the likelihood of observing $\mathbf{y}_t$ in state $\mathbf{x}_t$.

Particle filtering sequentially applies importance sampling to construct a particle set representation of $p(\mathbf{x}_t|\mathbf{y}_{1:t})$, the posterior density of state $\mathbf{x}_t$ given the sequence of observations $\mathbf{y}_{1:t} = (\mathbf{y}_1, \dots, \mathbf{y}_t)$, from a particle set representation of $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$. In its standard form (bootstrap filtering), the importance function and weights are

$$\mathbf{x}_t^{(j)} \sim p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) \approx \textstyle\sum_i w_{t-1}^{(i)} p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)})$$
$$w_t^{(j)} \propto p(\mathbf{y}_t|\mathbf{x}_t^{(j)}),$$

where the weights are normalized to sum to one.

2.2. BraMBLe Likelihood

In this section, we briefly describe the Bayesian Multiple Blob (BraMBLe) tracker; see [7] for details. The BraMBLe tracker provides a solution to tracking multiple occluding blobs/objects from video. The novelty of BraMBLe is its robust and natural multitarget observation likelihood model. BraMBLe searches for K blobs such that all pixels inside the blobs look like foreground and all pixels outside the blobs look like background, using learned models of the foreground and background appearance.

Given the hypothesized blob positions $\mathbf{x}$, BraMBLe first computes the label $l_g(\mathbf{x})$ for each image location g on a grid as either foreground (within *some* blob) or background (outside *all* blobs). The likelihood of the observed features $\mathbf{y}_g$ at grid point g given the label is modeled as a Gaussian mixture model (a separate background GMM is learned for each grid location, a common GMM is learned for the foreground). The grid features are assumed to be independent given the state, so the likelihood of the entire frame is $p(\mathbf{y}|\mathbf{x}) = \prod_g p_g(\mathbf{y}_g|l_g(\mathbf{x}))$, where $p(\mathbf{y}_g|fore)$ and $p_g(\mathbf{y}_g|back)$ are the GMMs.

The features $\mathbf{y}_g$ consist of six values. For each of the three opponent channels, we combine pixels in a small window around grid point g using a Gaussian and LoG filters at a set scale, as in [7]. The LoG filter is useful in differentiating the smooth mouse texture from the variable bedding texture, which are similar in color. The only noticeable failure for this choice of features is the mouse's shadow on the bedding. Figure 3(a) shows the log-likelihood ratio of foreground over background for some example frames.

The space searched by BraMBLe does grow exponentially with the number of targets. However, its likelihood function is smooth and well-behaved, in comparison to those used for contour tracking (Section 2.3). In addition, the number of parameters describing the blob is minimal. These properties make the search simple and robust.

Empirically, BraMBLe performs well at estimating the mouse positions (up to a permutation of identity labels). The main failure is that our blob representation of the mouse state (an ellipse) is not detailed or exact enough. First, for our application of automatic behavior analysis, it is beneficial to compute detailed model fits. Second, the approximate model makes the likelihood higher for some reasonable fits than others based on the relationship between the blob model and the true object shape (in the common case that the mouse is not precisely elliptical). Because of the grid independence assumptions, some reasonable fits score orders of magnitude higher than others. This is evident during occlusions, when the set of reasonable blob fits is large. BraMBLe greedily gives one fit a disproportionately large weight early in the occlusion. The particle sets produced are thus extremely sparse, and BraMBLe cannot later recover from its mistake.

2.3. MacCormick et al.'s Contour Likelihood

We combine the blob likelihood from BraMBLe with the "generic contour likelihood" described in [12]. Contour tracking searches for a contour model that matches edges in the image. The likelihood we use models the edges detected in the video frame along a sparse set of short measurement lines normal to and centered on the contour. The intersection between the hypothesized contour model and a measurement line is the center of the measurement line.

Briefly, we describe the measurement line likelihood model $p(\mathbf{y}_m|\mathbf{x})$; see [12, 2] for details. Let $\mathbf{y}_m$ be the be the binary vector indicating where edges are detected on measurement line m, n be the number of edges detected, and L be the measurement line length. If $n = 0$, then the measurement line likelihood is $p(\mathbf{y}_m|\mathbf{x}) = p_{01}$. If $n \geq 1$, with probability $1 - p_{01}$, one of these edge detections was produced by the hypothesized intersection, and the rest were produced by background clutter. With probability p_{01}, they were all produced by background clutter. The location of the edge produced by the hypothesized intersection is Gaussian around the line center with variance σ^2. Clutter edge detections are uniformly distributed along the line. The number of clutter detections follows a Poisson distribution with parameter λ. Thus, for $n_m \geq 1$,

$$p(\mathbf{y}_m, n|\mathbf{x}) = (1-p_{01})\frac{b(n-1)}{L^{n-1}}\sum_{i=1}^{n} y_{mi}N(i; L/2, \sigma^2)+p_{01}\frac{b(n)}{L^n}.$$

The measurement line observations are assumed independent given the state, so the total contour likelihood is $p(\mathbf{y}|\mathbf{x}) = \prod_m p(\mathbf{y}_m|\mathbf{x})$.

We found that, even for one mouse, the search performed by contour tracking was much more difficult than that performed by BraMBLe for multiple mice. In our occlusion-free training sequence of 300 frames, the generic contour tracker lost one of three mice completely. This is first because the contour model is of higher dimension, but mostly because the observation likelihood is much less smooth than the blob likelihood; if a pair of contours are not extremely close to the true fit, then the ranking given to the contours is often not meaningful. However, for contours that are close enough, the likelihood is usually peaked around a meaningful fit. This is in contrast to BraMBLe, in which the rankings are usually meaningful on a large scale but not a small scale. Even for one object, low-level blob tracking is useful for guiding the contour search [6]. Thus, while there is an explicit multitarget contour likelihood [13], we found that it alone did not work well for our data. We include contour tracking firstly because it estimates a more detailed position and secondly because there is more contour signal available during occlusions, both in the silhouette of the occlusion and the boundary between mice.

3. Blob and Contour Model

We use a particle filtering algorithm to approximate $p(\mathbf{x}_t|\mathbf{y}_{1:t})$, the posterior distribution of the state of all K mice in frame t, $\mathbf{x}_t = \mathbf{x}_{t,1:K}$, given blob and contour observations for frames 1 to t. In this section, we describe the combined blob and contour model we assume. In Section 3.1, we describe our model of the position of the mouse. In Section 3.2, we describe the observation likelihood. In Section 3.3, we describe the transition distribution.

3.1. Blob-Contour State

We represent the blob state of one mouse by an ellipse, parameterized by five shape and three velocity parameters: the center coordinates (μ_x, μ_y), the semimajor and semiminor axis lengths a and b, the rotation after scaling θ, the center velocity (v_x, v_y), and the semimajor axis velocity v_a,

$$\mathbf{x}_{bk} = (\mu_{xk}, \mu_{yk}, a_k, b_k, \theta_k, v_{xk}, v_{yk}, v_{ak})^\top.$$

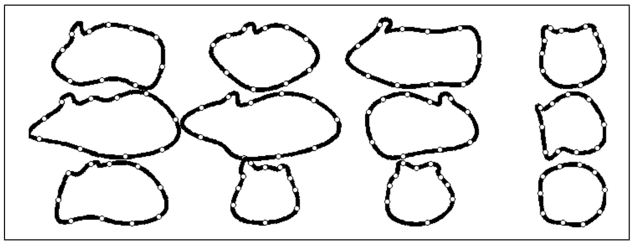


Figure 1. The 12 B-spline contour templates chosen to represent a mouse contour. The circles are locations of measurement lines.

We represent the contour state of one mouse by any affine transformation of any of the 12 B-spline templates shown in Figure 1. As the shape of the 2D mouse image is highly variable, there is no natural sense of correspondence between many pairs of templates. Thus, we cannot linearly combine contours, preventing the use of learned contour models such as [2]. We plan to explore methods for contour model learning in future work. This representation has the an application advantage over more flexible models such as [10, 8]: we can annotate the templates with e.g. ear, nose, or eye positions, useful for behavior analysis.

The locations of the measurement lines along each contour template are set by hand (see Figure 1). In addition, the minimum and maximum allowed eccentricity, pre-scaling rotation, and post-scaling rotation are set for each contour (no limits on post-scaling rotation are given for mice hanging from the ceiling). Each contour is also labeled as facing left, right, or both left and right. We parameterize the contour state by the eight ellipse parameters, the rotation before scaling ϕ, the contour template c, and whether the template is flipped f: $\mathbf{x}_k = (\mathbf{x}_{bk}, \phi_k, c_k, f_k)^\top$. The state of all K mice is the concatenation $\mathbf{x}_{1:K} = (\mathbf{x}_1^\top \dots \mathbf{x}_K^\top)^\top$.

3.2. Blob-Contour Observation Likelihood

We represent the useful information in an image observation by two sets of the features. The blob features $\mathbf{y}_b$ are those used by the BraMBLe tracker to determine how much each location matches the foreground and background models. The contour features $\mathbf{y}_c$ are the edges in the image. This representation encompasses much of the available signal, as the interior of the mouse is nearly featureless, preventing the use of image maps (e.g. [4, 1, 9]) or feature trackers [18]. Our current model ignores optical flow features [11, 22]. These features are useful in the mouse tracking domain [3], and we plan on incorporating them in the future.

To combine the BraMBLe likelihood $p(\mathbf{y}_b|\mathbf{x})$ and a soft version of the generic contour likelihood $p(\mathbf{y}_c|\mathbf{x}) = \prod_k p(\mathbf{y}_{ck}|\mathbf{x})$ (see Section 3.2.1), we assume conditional independence given the state of the mice:

$$p(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}_b|\mathbf{x}) \prod_{k=1}^{K} p(\mathbf{y}_{ck}|\mathbf{x}_k).$$

ICondensation[6] avoids this assumption by using a blob posterior (note that this posterior is not BraMBLe) as the importance function for contour tracking. It is essential that $p(\mathbf{y}_b|\mathbf{x})$ be included in our likelihood in order to use BraMBLe to model the multitarget dependencies.

3.2.1. A Soft Contour Likelihood

Contour tracking relies on an accurate edge detection algorithm. This is a challenge for our video sequence because there is a large amount of clutter in the scene. Much of the bedding has a high image intensity gradient and there are scratches on the cage. It is difficult to detect edges between the mice and bedding because the bedding in the shadow of the mice is very similar in color to the mouse. In addition, the edges between pairs of mice are subtle, if visible at all. We tried numerous edge detection methods, and the only one that gave reasonable results was the boundary detection algorithm used by the Berkeley Segmentation Engine (BSE) [14]. BSE computes the posterior probability of an edge based on the brightness, texture, and color gradient using a classifier trained on 12,000 manually labeled images. We credit BSE's superior performance in part to the texture gradient, which is robust to the types of clutter described. Figure 3(b) shows example images illustrating BSE's performance. The major downside of the BSE boundary detector is it is expensive – processing one entire image took over five minutes on a 2.8 GHz machine. We hypothesize that this algorithm can be optimized for tracking applications to reduce this cost.

Because the BSE boundary detector outputs meaningful probabilities rather than hard edge classifications, we used a soft version of the generic contour likelihood. This was essential for detecting edges between pairs of mice, as BSE often outputs a weak response for these edges (see Figure 3(b)). The BSE output for location i is $p(edge|\mathbf{y}_i)$, the probability that i is an edge given the observations $\mathbf{y}_i$. We model the probability of a binary classification of each measurement line pixel $\mathbf{z}$ given the edge features $\mathbf{y}$ as:

$$p(\mathbf{z}|\mathbf{y}) = \prod_{i=0}^{L} p(edge|\mathbf{y}_i)^{z_i}(1 - p(edge|\mathbf{y}_i))^{1-z_i}.$$

We assume equal priors for all $\mathbf{z}$, so this is equal to $p(\mathbf{y}|\mathbf{z})$. The probability of observing measurement line $\mathbf{y}$ given the hypothesized contour is the sum over all these possibilities:

$$p(\mathbf{y}|\mathbf{x}) = \sum_{z \in \{0,1\}^{L+1}} p(\mathbf{y}|\mathbf{z})p(\mathbf{z}, n|\mathbf{x}),$$

where $p(\mathbf{z}, n|\mathbf{x})$ is the generic contour likelihood described in Section 2.3. While this computation is extremely fast for small L, it grows exponentially with L. To combat this, the sum can be taken only over $\mathbf{z}$ such that $p(\mathbf{y}|\mathbf{z})$ is significant.

3.3. Motion Model

We make standard, simple assumptions in our model of the transition distribution $p(\mathbf{x}_t|\mathbf{x}_{t-1})$. First, we assume that the mice move independently, so we can factor $p(\mathbf{x}_{t,1:K}|\mathbf{x}_{t-1,1:K}) = \prod_k p(\mathbf{x}_{tk}|\mathbf{x}_{t,k-1})$. Second, all continuous shape parameters $\mathbf{x}_s = (\mu_x, \mu_y, a, b, \theta, \phi)$ follow independent constant velocity and/or autoregressive Gaussian diffusion models:

$$\mathbf{x}_{st}|\mathbf{x}_{t-1} \sim \mathcal{N}((\mathbf{I} - \mathbf{\Gamma})(\mathbf{x}_{s,t-1} + \mathbf{\Lambda}\mathbf{v}_{s,t-1}) + \mathbf{\Gamma}\hat{\mathbf{x}}_s, \mathbf{\Sigma}_s),$$

where $\mathbf{\Gamma}$ is the diagonal autoregressive constant (0 for μ_x, μ_y, and θ), $\mathbf{\Lambda}$ is the diagonal dampening constant (0 for b, θ, and ϕ), and $\mathbf{\Sigma}_s$ is the assumed diagonal covariance matrix. Velocities are set by subtracting the previous state from the current state.

There is a high probability that the contour template does not change. The probability of changing to a different contour template is based on whether the current and new contour face the same direction. We first determine which contours are allowed given the generated shape. If no contour is allowed, we rotate the shape after scaling by the minimum amount to allow at least one contour. We then decide which direction (left or right) the generated contour should face (if contours facing both directions are allowed). We flip the direction with probability proportional to the squared eccentricity. Given the direction, we choose an allowed contour facing that direction. If the direction has not changed, we choose the same contour with high probability. All other contours allowed in a given direction are given equal weight.

4. Blob-Contour Particle Filtering

In this section, we describe our algorithm for efficiently sampling from the combined blob and contour posterior distribution of the K mice, $p(\mathbf{x}_{tk}|\mathbf{y}_{b,1:t}, \mathbf{y}_{c,1:t})$, given the models and independence assumptions described in Section 3.

At each iteration of particle filtering, we generate a set of weighted samples $\{(x_t^{(i)}, w_t^{(i)})\}_{i=1}^N$ such that $p(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx \sum_{i=1}^N w_t^{(i)}\delta(\mathbf{x}_t - \mathbf{x}_t^{(i)})$ from the previous set of weighed samples $\{(x_{t-1}^{(i)}, w_{t-1}^{(i)})\}_{i=1}^N$. The most popular particle filtering algorithm is the bootstrap filter. As described in Section 2.1, the importance function used in this algorithm is $p(\mathbf{x}_{t,1:K}|\mathbf{x}_{t-1,1:K})$, and the importance weight is $p(\mathbf{y}_{bt}, \mathbf{y}_{ct}|\mathbf{x}_{t,1:K})$. Three properties of our problem cause this direct application to fail for any practical number of samples N. First, because the motion of the mice is erratic, the variance of $p(\mathbf{x}_{t,1:K}|\mathbf{x}_{t-1,1:K})$ is high. Second, because the contour feature is sparse, the scores given by $p(\mathbf{y}_{ct}|\mathbf{x}_{t,1:K})$ are only meaningful within a short radius of the optimal fits for all K mice. Third, the dimension of $\mathbf{x}_{t,1:K}$ is proportional to K, thus the search space size is exponential in K. Because of these three properties of the relationship between the importance and true posterior distributions, a huge number of samples must be generated so that enough fall within the short radius of the optimal fits.

We address these problems by performing three sampling steps, instead of just one. Each step incorporates a subset of the observations to gradually hone in on the small subspace truly important in the posterior distribution. The first step performs bootstrap filtering using only the blob observations. This step localizes the search space for the mice by moving the distribution toward the optimal fits and decreasing its variance. The second step performs bootstrap filtering using the contour observations independently for each mouse. This allows the algorithm to find the important regions for each mouse independently, exponentially reducing the search space by preventing a good fit for one mouse from being rejected because it is paired with a bad fit for another (this problem is also addressed in the sensor observation literature [15]). The third step of sampling combines the K sets of particles by sampling independently from each, then weights by the necessary importance weight.

One can interpret the first two filtering steps as generating samples from the posterior marginals. Then, the importance function in the third sampling step is the product of the posterior marginals:

$$q_3(\mathbf{x}_t|\mathbf{y}_{1:t}) = \prod_{k=1}^{K} p(\mathbf{x}_{tk}|\mathbf{y}_{1:t-1,bt,ctk}).$$

We describe these three steps in more detail next. Figure 2 shows the algorithm steps.

```
For $t = 1, 2, \ldots$:
 1. Sample from the marginal posteriors:
    For $i = 1, \ldots, N$:
      a. Choose $\tilde{\mathbf{x}}_{t-1,1:K}^{(i)} \sim \{(\mathbf{x}_{t-1,1:K}^{(j)}, w_{t-1}^{(j)})\}$.
      b. Generate $\mathbf{b}_{1:K}^{(i)} \sim p(\mathbf{b}_{1:K}|\tilde{\mathbf{x}}_{t-1,1:K}^{(i)})$.
      c. Compute the weight $w_b^{(i)} \propto p(\mathbf{y}_{bt}|\mathbf{b}_{1:K}^{(i)})$.
      d. Choose $\tilde{\mathbf{b}}_{1:K}^{(i)} \sim \{\tilde{\mathbf{b}}_{1:K}^{(j)}, w_b^{(j)})\}$
      e. For $k = 1, \ldots, K$,
         Generate $\tilde{\mathbf{x}}_{tk}^{(i)} \sim p(\mathbf{x}_{tk}|\tilde{\mathbf{b}}_k^{(i)})$.
      f. Compute the weight $w_{tk}^{(i)} = p(\mathbf{y}_{ctk}|\tilde{\mathbf{x}}_{tk}^{(i)})$.
 2. Sample from the joint posterior:
    For $i = 1, \ldots, N$:
      a. For $k = 1, \ldots, K$,
         Choose $\mathbf{x}_{tk}^{(i)} \sim \{(\tilde{\mathbf{x}}_{tk}^{(j)}, w_{tk}^{(j)})\}$.
      b. Concatenate: $\mathbf{x}_{t,1:K}^{(i)} = (\mathbf{x}_{t1}^{(i)}, \ldots, \mathbf{x}_{tK}^{(i)})$.
      c. Compute the importance weight:
         $w_t^{(i)} \propto p(\mathbf{y}_{bt}|\mathbf{x}_{t,1:K}^{(i)}) \sum_{j=1}^{N} w_{t-1}^{(j)} \prod_{k=1}^{K} p(\mathbf{x}_{tk}^{(i)}|\mathbf{x}_{t-1,k}^{(j)})$.
```

Figure 2. Blob and Contour Particle Filtering

Step 1: Blob Sampling

In the first sampling step, we perform an iteration of bootstrap filtering (with systematic resampling with replacement) using only the blob observation $\mathbf{y}_{bt}$. Given $\{(\mathbf{x}_{t-1,1:K}^{(i)}, w_{t-1}^{(i)})\}$ representing $p(\mathbf{x}_{t-1,1:K}|\mathbf{y}_{1:t-1})$, we generate $\{\mathbf{b}_{t,1:K}^{(i)}\}_{i=1}^{N}$ from

$$q_1(\mathbf{b}_t|\mathbf{y}_{1:t}) = \int p(\mathbf{b}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})d\mathbf{x}_{t-1},$$

where $p(\mathbf{b}_t|\mathbf{x}_{t-1}) = \prod_k p(\mathbf{b}_{tk}|\mathbf{x}_{t-1,k})$ and $p(\mathbf{b}_{tk}|\mathbf{x}_{t-1,k})$ has the same form as $p(\mathbf{x}_{tk}|\mathbf{x}_{t-1,k})$ for all blob parameters, and is a δ function around the previous state for the contour parameters. We then weight by $w_{bt}^{(i)} \propto p(\mathbf{y}_{bt}|\mathbf{b}_t)$ so that

$$p(\mathbf{b}_t|\mathbf{y}_{1:t-1,bt}) \approx \sum_i w_{bt}^{(i)} \delta(\mathbf{b}_t - \mathbf{b}_t^{(i)}).$$

The marginal is

$$p(\mathbf{b}_{tk}|\mathbf{y}_{1:t-1,bt}) = \int p(\mathbf{b}_t|\mathbf{y}_{1:t-1,bt})d\mathbf{b}_{t1:K\backslash k}$$

$$\approx \int \sum_i w_{bt}^{(i)} \delta(\mathbf{b}_t - \mathbf{b}_t^{(i)})d\mathbf{b}_{t,1:K\backslash k} = \sum_i w_{bt}^{(i)} \delta(\mathbf{b}_{tk} - \mathbf{b}_{tk}^{(i)}).$$

Step 2: Independent Contour Sampling

In the second sampling step, we perform an iteration of bootstrap filtering using the contour observation $\mathbf{y}_{ctk}$ for each mouse k independently. Given $\{(\mathbf{b}_{tk}^{(i)}, w_{bt}^{(i)})\}$ representing $p(\mathbf{b}_{tk}|\mathbf{y}_{1:t-1,bt})$, we generate $\{\tilde{\mathbf{x}}_{tk}^{(i)}\}$ from

$$q_2(\mathbf{x}_{tk}|\mathbf{y}_{1:t}) = \int p(\mathbf{x}_{tk}|\mathbf{b}_{tk})p(\mathbf{b}_{tk}|\mathbf{y}_{1:t-1,bt})d\mathbf{b}_{tk},$$

where $p(\mathbf{x}_{tk}|\mathbf{b}_{tk})$ is the same as $p(\mathbf{x}_{tk}|\mathbf{x}_{t-1,k})$ for the contour parameters and Gaussian (with a different, smaller variance) around $\mathbf{b}_{tk}$ for the blob parameters. We then weight by $w_{tk}^{(i)} \propto p(\mathbf{y}_{ctk}|\tilde{\mathbf{x}}_{tk})$ so that

$$p(\mathbf{x}_{tk}|\mathbf{y}_{1:t-1,bt,ctk}) \approx \sum_i w_{tk}^{(i)} \delta(\mathbf{x}_{tk} - \tilde{\mathbf{x}}_{tk}^{(i)}).$$

Step 3: Combining the Marginals

In the final sampling step, we perform an iteration of sampling to combine the individual mouse marginals. Given $\{(\tilde{\mathbf{x}}_{tk}^{(i)}, w_{tk}^{(i)})\}$ representing $p(\mathbf{x}_{tk}|\mathbf{y}_{1:t-1,bt,ctk})$ for $k = 1, \ldots, K$, we generate $\{\mathbf{x}_t^{(i)}\}$ from

$$q_3(\mathbf{x}_{tk}|\mathbf{y}_{1:t}) = \prod_{k=1}^{K} p(\mathbf{x}_{tk}|\mathbf{y}_{1:t-1,bt,ctk})$$

by independently sampling from $\{(\tilde{\mathbf{x}}_{tk}^{(i)}, w_{tk}^{(i)})\}$ for $k = 1, \ldots, K$ and concatenating. We then weight by

$$\begin{aligned}
w_t^{(i)} &\propto \frac{p(\mathbf{x}_t^{(i)}|\mathbf{y}_{1:t})}{q_3(\mathbf{x}_t|\mathbf{y}_{1:t})} \\
&\propto \frac{p(\mathbf{y}_{bt}|\mathbf{x}_t^{(i)}) \prod_k p(\mathbf{y}_{ctk}|\mathbf{x}_{tk}^{(i)}) \int p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})d\mathbf{x}_{t-1}}{\prod_k p(\mathbf{x}_{tk}|\mathbf{y}_{1:t-1,bt,ctk})} \\
&\approx \frac{p(\mathbf{y}_{bt}|\mathbf{x}_t^{(i)}) \prod_k p(\mathbf{y}_{ctk}|\mathbf{x}_{tk}^{(i)}) \int p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}) \sum_j w_{t-1}^{(j)} \delta(\mathbf{x}_{t-1} - \mathbf{x}_{t-1}^{(j)})d\mathbf{x}_{t-1}}{\prod_k \sum_j w_{tk}^{(j)} \delta(\mathbf{x}_{tk}^{(i)} - \tilde{\mathbf{x}}_{tk}^{(j)})} \\
&= \frac{p(\mathbf{y}_{bt}|\mathbf{x}_t^{(i)}) \prod_k p(\mathbf{y}_{ctk}|\mathbf{x}_{tk}^{(i)}) \sum_j w_{t-1}^{(j)} p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(j)})}{\prod_k p(\mathbf{y}_{ctk}|\mathbf{x}_{tk}^{(i)})} \\
&= p(\mathbf{y}_{bt}|\mathbf{x}_t^{(i)}) \sum_j w_{t-1}^{(j)} p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(j)})
\end{aligned}$$

Our sampling algorithm is consistent because it is a sequence of three consistent steps. The first two steps are standard applications of bootstrap filtering, and are thus consistent. So, each of the K particle sets input in step 3 in the limit as $N \to \infty$ converge pointwise to the true marginal posterior. Step 3 generates samples from the product of the marginal posteriors. If any of the marginals has zero density, then the joint density is also zero. Finally, we weight by the ratio of the joint posterior to the particle approximations of the marginal posteriors. As these particle approximations are consistent, the ratio is also consistent.

5. Comparison to Related Work

A number of algorithms address the shortcomings of bootstrap filtering for multitarget tracking. One set incorporates information from the current frame in the importance function. The other modifies the algorithm to perform sampling independently for each target. Our algorithm performs both of these modifications, incorporating the blob observation and sampling from the target marginals.

Within the first category falls [16], which incorporates object detection [21] in the importance function, which is similar to incorporating the blob observation. ICondensation [6] uses a single-blob observation in the importance function, but this algorithm is only a contour tracker (the effect of the blob observation is divided out), and is not a multitarget algorithm. The unscented particle filter [19] uses the unscented Kalman filter to incorporate the current observation into the importance function. We found that this performed poorly on our data because our likelihood function is extremely nonlinear.

The problems with tracking many targets is often addressed in applications in which observations are from non-visual sensors such as radar. The most popular approach is the Joint Probabilistic Data Association Filter (JPDAF) [17], which estimates the product of the marginal posterior distributions instead of the joint posterior distribution. A number of algorithms similar to the JPDAF are described in [20]. Algorithms from this literature are not directly applicable to our application because our observation likelihoods are more complicated, and do not factor over targets. In the contour tracking literature, partitioned sampling [12] also addresses the multitarget dimensionality issue by first estimating the position of the target in front, then using this estimate to determine the position of the second target, and so on. However, this algorithm uses only the contour likelihood, which is a local measure for the image, allowing them to compute the likelihood given the states of any number of targets, while the BraMBLe likelihood requires the states of all targets. In addition, the state of the front target is resampled as the state of each of the rest of the targets is resampled, thus it is resampled K times in each iteration, leading to sample depletion.

6. Experiments

We evaluated our blob and contour tracking algorithm on a video sequence of three identical mice exploring a cage, available at `http://smartvivarium.calit2.net`. This sequence contained 11 occlusions of varying difficulty. The model parameters were chosen by hand using a separate video sequence. Many of the parameters were set using our knowledge about the problem. These include the variance of the transition models and the constraints on the state. Other parameters, including the damping and autoregressive constants, were set to values used in [2]. The contour likelihood parameters were set so that the ranking of the probability of each vector of observed edge detections seemed reasonable. Some of the parameters were chosen somewhat arbitrarily and never varied – these include the number of measurement lines and the parameters used in the BraMBLe likelihood. The number of samples was chosen to be $N = 2000$. While we had qualitatively similar performance with $N = 1000$ samples, the results returned by particle filtering varied quite a bit. We thus chose to present results with $N = 2000$ samples, for which the output of our particle filtering algorithm was stable. This number of samples compares favorably to the 4000 samples used to track a pair of leaves in [13] and the 1000 samples used to track two people/blobs in [7].

We provide with this paper a video of our results. Summary still frames are shown in Figure 4. These results demonstrate the following strengths of our algorithm:

- Our contour tracking algorithm is robust to erratic mouse behavior – we never lose a mouse. For instance, we follow mice that jump, drop from the ceiling, and make quick turns and accelerations that are not fit by our simple dynamics model (see Figure 4(a)).
- Two contours never fit the same mouse.
- Our algorithm is rarely distracted by background clutter. This implies that our feature extraction methods and the blob and contour combination provide robust observation likelihoods. The only exceptions are when *both* algorithms make mistakes: when the blob tracker mistakes shaded bedding for foreground and the contour tracker fits to the edge of a tail (see Figure 4(b) for an example).
- Perhaps the most impressive result is that our algorithm accurately tracks the mice through 7 out of 11 occlusions and partway through the other 4. This is because of the detailed fit provided by the contour tracking algorithm and its ability to use features available during occlusions. Example successful frames are shown in Figure 4(c).
- In general, our algorithm usually found very good contour fits outside of occlusions, much better than those obtained using contour tracking alone.

Our algorithm has a couple of failure modes which we plan on addressing in future work. First, it occasionally gets stuck in local optima in which the contour fit was facing the wrong direction (see Figure 4(b)). We plan to address this problem with a better model of the probability of the direction changing. Second, our algorithm swaps identity labels in four occlusions (see Figure 4(d)). The reason for this is that the fit of our algorithm is heavily biased by the fit of the BraMBLe algorithm. For occlusions in which the contour observation signals are weak, this bias from BraMBLe can dominate. We propose a solution to this in Section 7.

For comparison, we also implemented a combined blob-contour tracking algorithm which performed two steps of sampling instead of three, resulting in the final importance function

$$q'(\mathbf{x}_t|\mathbf{y}_{1:t}) = \int p(\mathbf{x}_t|\mathbf{b}_t)p(\mathbf{b}_t|\mathbf{y}_{1:t-1,bt})d\mathbf{b}_t.$$

This algorithm has the disadvantage that samples of all k mice are weighted by the product of the contour likelihood for each mouse. Thus, the number of samples fitting blobs is the same, but the effective number of samples used when fitting contours is much smaller. We tested this algorithm on 500 frames with 6 occlusions (our algorithm works on 4). The results fit our theory. While this algorithm was resistant to drift (blob tracking is the same in both algorithms), the contour fits found were less satisfactory. In general, they were less fine-tuned and more variable from frame to frame, suggesting that more samples are needed. This is particularly evident during occlusions. The fits during occlusions are less fine-tuned to the contour data, and therefore more influenced by the blob tracking results. This causes worse fits in every occlusion in the sequence. This algorithm swapped identities twice more than the algorithm proposed in this paper.

7. Conclusions and Future Work

Our algorithm combines the BraMBLe likelihood [7] with the "generic contour likelihood" [12] to utilize a robust set of features necessary for our noisy, real world mouse video. Our algorithm breaks each iteration of particle filtering into three steps, each incorporating a subset of the observations to gradually hone in on the small space in which most mass of the posterior density lies. This can also be seen as modifying the importance function of the particle filtering algorithm to be the product of the posterior marginal densities, thus incorporating the current observation in the importance function. This dramatically reduces the number of samples necessary for accurate tracking.

In future work, we plan on exploring algorithms that use information from both the past and the future to determine the positions of the mice during an occlusion. We hope that this will solve the main failure of the algorithm proposed in this algorithm by making BraMBLe return a more global fit. We are exploring heuristic solutions to this problem, in the direction of [3].

Acknowledgments

We would like to thank Sameer Agarwal, Sanjoy Dasgupta, Piotr Dollar, Ben Ochoa, Vincent Rabaud, Eric Wieriora, and Josh Wills for invaluable comments and discussion, Keith Jenne, Phil Richter, Geert Schmid-Schoenbein, and John Wesson for providing us with data and mouse-knowledge, and Craig Donner and Henrik Wann Jensen for use of their machines. This work is funded by NASA GSRP grant NGT5-50479, the UCSD Division of Cal-(IT)2 - the California Institute for Telecommunications and Information Technology - under the Smart Vivarium project, NSF Career grant #0448615, subcontracts B542001 and B547328 under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract No. W-7405-ENG-48, and the Alfred P. Sloan Fellowship.

References

[1] B. Bascle and R. Deriche. Region tracking through image sequences. In *ICCV*, pages 302–307, 1995.

[2] A. Blake and M. Isard. *Active Contours*. Springer, Great Britain, 2000.

[3] K. Branson, V. Rabaud, and S. Belongie. Three brown mice: See how they run. In *VS-PETS Workshop at ICCV*, 2003.

[4] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *PAMI*, 25(5):564–575, 2003.

[5] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.

[6] M. Isard and A. Blake. ICONDENSATION: Unifying low-level and high-level tracking in a stochastic framework. *Lecture Notes in Computer Science*, 1406:893–908, 1998.

[7] M. Isard and J. MacCormick. BraMBLe: A Bayesian multiple-blob tracker. In *ICCV*, 2001.

[8] J. Jackson, A. Yezzi, and S. Soatto. Tracking deformable moving objects under severe occlusions. In *IEEE Conf. on Decision and Control*, 2004.

[9] N. Jojic and B. Frey. Learning flexible sprites in video layers. In *CVPR*, 2001.

[10] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *IJCV*, pages 321–331, 1988.

[11] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *DARPA Image Understanding Workshop*, 1981.

[12] J. MacCormick. *Stochastic Algorithms for Visual Tracking*. Distinguished Dissertations. Springer, Great Britain, 2002.

[13] J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. *IJCV*, 39(1):57–71, 2000.

[14] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(5):530–549, '04.

[15] W. Ng, J. Reilly, and T. Kirubarajan. A bayesian approach to tracking wideband targets using sensor arrays and particle filters. In *Signal Processing*, 2003.

[16] K. Okuma, A. Taleghani, N. de Freitas, J. Little, and D. Lowe. A boosted particle filter: Multitarget detection and tracking. In *ECCV*, 2004.

[17] Y. B. Shaalom and T. E. Fortman. *Tracking and Data Association*. Academic Press, Boston, 1988.

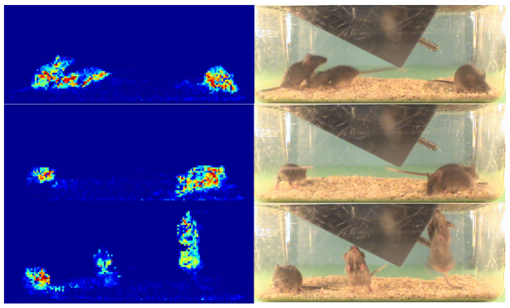
[18] J. Shi and C. Tomasi. Good features to track. In *CVPR*, 1994.

[19] R. van der Merwe, A. Doucet, N. de Freitas, and E. Wan. The unscented particle filter. In *NIPS*, 2000.

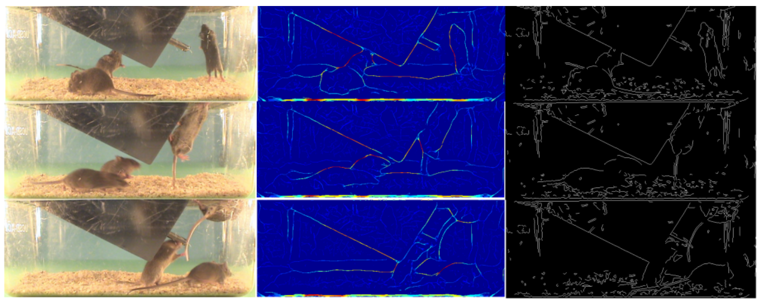
[20] J. Vermaak, S. Godsill, and P. Perez. Monte carlo filtering for multi-target tracking and data association. In *Tr. Aerospace*, 2005.

[21] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.

[22] Y. Weiss and E. Adelson. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *CVPR*, pages 321–326, 1996.

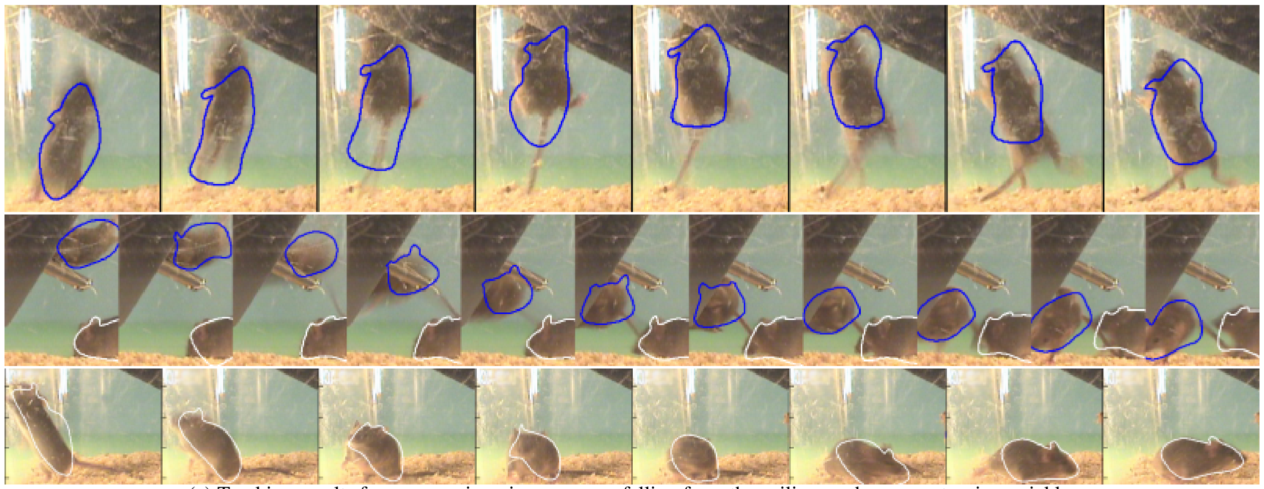


(a) The log-likelihood ratio of foreground over background for selected frames. Image ii shows an occlusion. Image iii shows the tail and shadow of the mice.

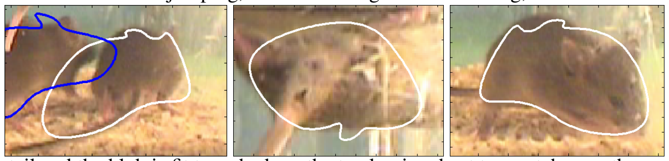


(b) The second column shows the BSE output; the third shows the canny edge detector's output. In image i an edge between a pair of mice is found. In image ii, BSE gives a weak response to an edge between a pair of mice. In image iii, BSE is robust to scratches.

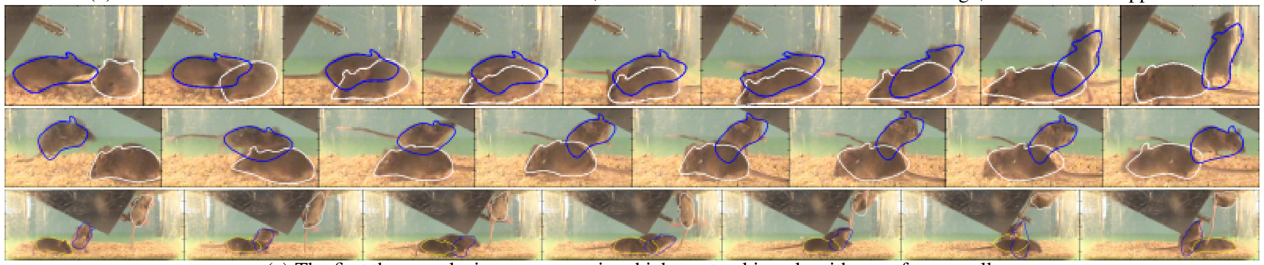
Figure 3. Features extracted for the (a) blob and (b) contour observation likelihoods.



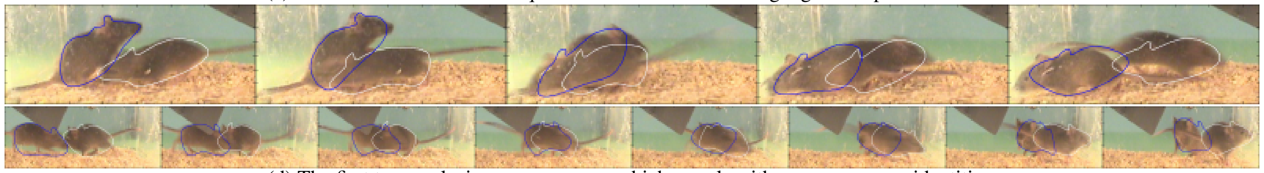
(a) Tracking results for a mouse jumping, a mouse falling from the ceiling, and a mouse turning quickly.



(b) The contour is fit to a tail and the blob is fit to a shadow; the tracker is robust to scratches on the cage; the contour is flipped.



(c) The first three occlusion sequences in which our tracking algorithm performs well.



(d) The first two occlusion sequences on which our algorithm swaps mouse identities.

Figure 4. Still frame summary of our results. We plot the average affine transformation applied to the contour with the most total weight.