

Tracking Multiple Indistinguishable Objects through Severe Occlusion

Abstract

Occlusions are one of the biggest obstacles to reliable tracking. While many existing algorithms perform well while the objects of interest are separated, they often swap object identities when multiple objects occlude one another. We focus on the problem of tracking multiple, nonrigid targets through severe occlusions. Our occlusion reasoning algorithm takes advantage of reliable separated object tracking before and after occlusion events. We use BraMBLe to track separated objects and determine the starts and ends of occlusions. From the separated object tracks, our algorithm uses a simple depth-order heuristic to guess the identity correspondence in the start and end frames of the occlusion. We interpolate the correspondence to yield a guess of the objects' per-frame motion. This motion guess is incorporated into layer-based affine optical flow estimation in the form of a prior probability. Our algorithm tracks the objects forward one frame at a time from the start of the occlusion and backward one frame at a time from the end of the occlusion. The motion guess is updated based on the current measurements of the object positions. These steps are repeated until the object tracks meet in the middle of the occlusion. We demonstrate results on challenging test video of three identical mice from a side view.

1. Introduction

Recently, a number of works have appeared that address the problem of multiple object (or blob) tracking [6, 1, 8]. These algorithms perform well while the objects of interest are separated. However, to solve the multiple object tracking problem, it is vital to track object identities through severe occlusions. To this end, existing approaches make assumptions about the targets. For example, some approaches require object-specific appearance models such as color histograms [1]. Others assume a constant velocity model of object motion [6]. In this paper, we focus on the occlusion reasoning problem when these assumptions break down: the case in which the objects are indistinguishable and their motions are erratic. Instead of relying on object model assumptions, we leverage reliable separated object tracking before and after occlusion events.

Using the tracks of the separated objects, we detect the start and end frames of an occlusion event. We use a heuristic to estimate which of the objects in the occlusion is clos-

est to the camera. As the objects cannot pass through one another during an occlusion event, their depth ordering cannot change, as observed in [7]. We thus associate the front object at the start of an occlusion with the front object at the end of the occlusion.

Using this correspondence, ideally we would find the object configuration in every frame of the occlusion that carries the front object at the start to the front object at the end and best fits the object models and image data. As performing this search is intractable, we approximate the best sequence of configurations sequentially while incorporating a hint of the future locations of the objects. Our approach interpolates the correspondence guess into a series of per-frame motion guesses. The predicted per-frame motion guess is incorporated into layer-based affine optical flow estimation in the form of a prior probability distribution.

For each pair of consecutive frames, our algorithm solves for the affine transformations that best describe the motion in each object layer under the influence of the per-frame motion guess. After the objects are tracked through a pair of frames, the per-frame motion guesses are updated based on the estimated positions of the objects. As we have available all frames during the occlusion event, we can choose the order in which we process the frames. We would like to process the pairs of frames that give the most reliable results first. We therefore first track the objects through the first and last pairs of frames in the occlusion, then the second and second-to-last pairs of frames, and so on.

We have tested our algorithm on a challenging video sequence of three indistinguishable mice exploring a cage. As our sequence is taken from the side of the cage, there is severe occlusion.

The organization of this paper is as follows. We describe our proposed approach in Section 2. In this Section, we describe the three components of our algorithm: separated object tracking, occlusion detection, and occlusion reasoning. In Section 3 we provide experimental results on real world footage of caged mice. We conclude and discuss future directions in Section 4.

A preliminary version of this approach is described in the workshop paper [2], but with a simpler and less robust separated object tracking module and without tracking symmetrically forward and backward in time during occlusion events.

2. Our Approach

Our approach breaks the tracking task into the subproblems of tracking separated objects, detecting occlusions, and reasoning about occlusions. Our algorithm begins by using the Bayesian Multiple Blob tracker (BraMBLe) to track the objects while they are separated [6]. In Section 2.1, we motivate our choice of the BraMBLe tracker. Using the object positions computed by BraMBLe, we check at each frame for the start of an occlusion. We discuss our occlusion detection heuristics in Section 2.2. When an occlusion is detected, we continue to use BraMBLe to track the objects, temporarily allowing identity swapping to occur, and checking at each frame for the end of an occlusion. When the end of an occlusion is detected, we re-estimate the positions of the objects in the occlusion using our novel occlusion reasoning algorithm, described in Section 2.3. A summary of our algorithm is shown in Figure 4.

2.1. Separated Object Tracking

Our occlusion reasoning module places several demands on our separated object tracking algorithm. First, it should give accurate position and identity results when the objects are separated. When the objects are occluding one another, it should be able to determine which objects are in the occlusion. Finally, it should be able to give accurate object position (but not object identity) results when the objects are slightly occluding one another.

Thus, while we only use the positions returned by the separated object tracker when the objects are separated, we require that the separated object tracker have the correct model of object occlusion. By this, we mean that a single location in the image can belong to multiple objects, but the pixel color at that location only reflects the appearance of the front object. This is in contrast to a Gaussian Mixture Model, for example, which assumes that each data point belongs to one Gaussian (the membership of a pixel to each Gaussian must sum to one).

Because of these requirements, we use the BraMBLe tracker [6]. BraMBLe is a particle filter with a multi-blob likelihood function. That is, it represents the probability of a configuration of objects given the image data with a particle set. This representation allows all the configuration of objects to be estimated together, instead of estimating the position of each object independently. In addition, the observation likelihood model used by BraMBLe correctly handles occlusions, in that a single location in the image can belong to multiple objects, but the pixel color at that location only reflects the appearance of the front object.

2.2. Detection of Occlusions

At each frame, we examine the positions estimated by BraMBLe to check for the start of an occlusion. When an

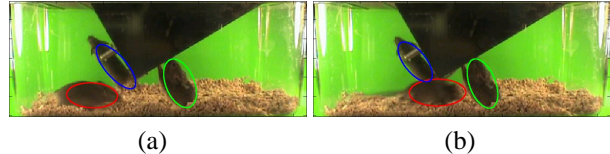


Figure 1: Motivation for two occlusion detection heuristics. The images shown depict three mice in a cage; the dark triangle at the top is the visible portion of the feeder box. (a) An example in which the Fisher distance in the x -direction between a pair of objects is small, but there is no occlusion. (b) An example in which the intersection of two objects is small, but there is an occlusion. Ending the occlusion at this point would result in incorrect depth estimates.

occlusion is detected, we continue tracking using BraMBLe through the occlusion, examining the positions estimated by BraMBLe to check for the end of the occlusion.

We combine two heuristics to determine if an object, m_1 , is occluding another object, m_2 . The first is the fraction of all the locations belonging to either object in the intersection of the objects. That is, if there are n_1 points in the first object, n_2 points in the second object, and n_{12} points in the intersection of the objects, we compute

$$d_I(m_1, m_2) = \frac{2n_{12}}{n_1 + n_2}.$$

The second heuristic is a measure of the x -distance between the two objects. We use the Fisher distance in the x -direction between the distributions of locations belonging to each object. If μ_{1x} and μ_{2x} are the x -coordinates of the means of the two distributions and σ_{1x}^2 and σ_{2x}^2 are the variances of the two distributions in the x -direction, then the Fisher distance is

$$d_F(\mu_{1x}, \sigma_{1x}^2, \mu_{2x}, \sigma_{2x}^2) = \frac{(\mu_{1x} - \mu_{2x})^2}{(\sigma_{1x}^2 + \sigma_{2x}^2)/2}.$$

We threshold both of these heuristics. If the intersection fraction is big enough and the Fisher distance is small enough, the objects are declared to be in an occlusion. In Figure 1, we show examples where individually one of these two heuristics would fail, but combined they correctly detect occlusions.

2.3. Occlusion Reasoning

Our occlusion reasoning algorithm is based on the ‘‘Direct Methods’’ [5] approach to layer-based affine optical flow estimation. Each object is represented in a 2-D layer, and the layers are ordered by depth. Given the pixels belonging to each layer/object in the first frame, we compute the ‘‘best’’ affine transformation describing the motion of each layer/object between the pair of frames. Given these affine

motion estimates, the pixels in the second frame belonging to each layer/object are segmented.

We describe our criterion for the “best” affine motion in Sections 2.3.1 and 2.3.2. In Section 2.3.4, we describe the segmentation of pixels into layers. In Section 2.3.5, we describe the order in which pairs of frames are processed.

2.3.1 Affine Flow Estimation

Our occlusion reasoning algorithm is based on optical flow estimation using multiple affine models. Consider the set of pixels belonging to one layer. We assume that the Horn-Schunck brightness constancy condition holds within this set of pixels, so that

$$I_x u + I_y v + I_t = 0$$

Here, $I(x, y, t) \in [0, 1]$ denotes the intensity at location $(x, y)^\top$ and time t , the subscript denotes partial differentiation and u and v are the x and y components of the flow at (x, y) . As in [5], we use an affine model for the flow of the form

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} a_1 + a_2 x + a_3 y \\ a_4 + a_5 x + a_6 y \end{pmatrix}.$$

In the least-squares sense, the best \mathbf{a} given only the optical flow cue minimizes

$$H_0[\mathbf{a}] = \sum_{(x,y) \in \mathcal{M}} w(x, y) (\mathbf{z}^\top \mathbf{a} + I_t)^2,$$

where \mathcal{M} is the set of pixels belonging to the object, the vectors \mathbf{z} and \mathbf{a} are defined as

$$\begin{aligned} \mathbf{z} &= (I_x, I_x x, I_x y, I_y, I_y x, I_y y)^\top, \\ \mathbf{a} &= (a_1, \dots, a_6)^\top \end{aligned}$$

and $w(x, y)$ is a measure of the certainty that pixel (x, y) at time t is in \mathcal{M} .

In severe occlusions, the optical flow cue alone is not enough to get an accurate motion estimate. We thus add a hint of the future object locations in the form of a quadratic regularization term, which nudges the estimate \mathbf{a} toward the prior affine motion estimate $\hat{\mathbf{a}}$, to be discussed in Section 2.3.2. We use the term “prior” because of the close relation of this regularization term to an assumed prior distribution on \mathbf{a} [4]. The strength of this nudge, for each component of \mathbf{a} , is defined by the 6×6 matrix $\lambda \Sigma_a^{-1}$. The scalar λ sets the weight of the regularization penalty relative to the optical flow estimate. The matrix Σ_a is a measure of the relative weights of the regularization for each of the individual entries of \mathbf{a} . We take Σ_a to be diagonal. Each entry corresponds to our guess of the amount of variance in the corresponding entry of \mathbf{a} . With this regularization term, our new criterion is

$$H[\mathbf{a}] = \sum_{(x,y) \in \mathcal{M}} w(x, y) (\mathbf{z}^\top \mathbf{a} + I_t)^2 + \lambda (\mathbf{a} - \hat{\mathbf{a}})^\top \Sigma_a^{-1} (\mathbf{a} - \hat{\mathbf{a}}).$$

Taking the partial derivative of $H[\mathbf{a}]$ with respect to \mathbf{a} , setting it to zero, and solving for \mathbf{a} , we find that

$$\mathbf{a} = (Z^\top W Z + \lambda \Sigma_a^{-1})^{-1} (-Z^\top W \mathbf{I}_t + \lambda \Sigma_a^{-1} \hat{\mathbf{a}}),$$

where Z is the $|\mathcal{M}| \times 6$ matrix with rows \mathbf{z}^\top , W is a $|\mathcal{M}| \times |\mathcal{M}|$ diagonal matrix of the weights w , and \mathbf{I}_t is a length $|\mathcal{M}|$ vector of the I_t .

Note the following special cases:

$$\mathbf{a} = -(Z^\top W Z)^{-1} Z^\top W \mathbf{I}_t \quad \text{as } \lambda \rightarrow 0$$

which optimizes $H_0[\mathbf{a}]$, and

$$\mathbf{a} = \hat{\mathbf{a}} \quad \text{as } \lambda \rightarrow \infty$$

in which the \mathbf{a} is chosen without regard to the optical flow computation.

Note that the affine motion is estimated only for pixels labeled as unoccluded, while we would like to estimate the motion of all pixel locations, occluded and unoccluded. It is only safe to assume that the affine transformation for the visible part of the object equals the affine transformation for the entire object if a significant portion of the object is observable. While the weight of the optical flow term in the form above is proportional to the number of unoccluded pixels, we found that this weight does not degrade fast enough. We thus ignore the optical flow estimate if more than some fraction (we chose 0.7) of the object is occluded, and rely only on the affine motion prior $\hat{\mathbf{a}}$.

2.3.2 Prior Estimation

Next, we discuss the choice of $\hat{\mathbf{a}}$ for each layer in each frame of the occlusion. As mentioned before, $\hat{\mathbf{a}}$ can be interpreted as the mean of the prior distribution on \mathbf{a} . The transformation $\hat{\mathbf{a}}$ is a guess of the per-frame motion of each object. It is derived from the estimated correspondence between objects at the start and end of an occlusion.

To estimate this correspondence, we use only the depth ordering cue, though other cues (e.g. individual appearance models, motion models) could be incorporated. The depth ordering cue is an estimate of which object is the front blob at the start of the occlusion and which blob is in front at the end of the occlusion. As these blobs must correspond to the same object, we reason that during the occlusion event, the succession of frame to frame motions must transform the mean and covariance statistics of the pixel locations of the initial front object to the mean and covariance statistics of the final front object. We interpolate the total transformation between the initial and final statistics of the front object into per-frame transformations.

We cannot assume that the back objects do not change depth ordering with respect to each other during the occlusion. Instead, we assume that the average mean and covariance statistics of the back objects at the start of the occlusion

correspond to the average mean and covariance statistics of the back objects at the end of the occlusion. Thus, for all the back objects, we calculate a single prior estimate from the locations of these average statistics. We set the prior estimates for each of the back objects to this common prior.

While many more sophisticated interpolations exist, we found that linearly interpolating the affine motion worked well. To describe the interpolation, we will break the affine motion into two parts,

$$A = \begin{pmatrix} a_2 & a_3 \\ a_5 & a_6 \end{pmatrix}, \quad \mathbf{t} = \begin{pmatrix} a_1 \\ a_4 \end{pmatrix}$$

If the displacement (u, v) is computed in the coordinate system centered on $\boldsymbol{\mu}$, then applying the affine transformation $\{A, \mathbf{t}\}$ to pixel locations with moments $\{\boldsymbol{\mu}, \Sigma\}$ will result in pixel locations with moments

$$\boldsymbol{\mu}' = \boldsymbol{\mu} + \mathbf{t}, \quad \Sigma' = A\Sigma A^\top.$$

We first compute the transformation $\{A_{1:n}, \mathbf{t}_{1:n}\}$ that transforms the statistics of the object in the first frame of the occlusion event, $\{\boldsymbol{\mu}_1, \Sigma_1\}$ to the statistics of the object in the last frame of the occlusion event, $\{\boldsymbol{\mu}_n, \Sigma_n\}$, where n is the number of frames in the occlusion event. Any pair in the family

$$\mathbf{t}_{1:n} = \boldsymbol{\mu}_n - \boldsymbol{\mu}_1 \\ A_{1:n} = \Sigma_n^{1/2} O^\top \Sigma_1^{-1/2}$$

where O is an arbitrary orthogonal matrix will perform the desired transformation [3]. We set the matrix O equal to the identity because the next step requires $A_{1:n}$ to be positive semidefinite. We estimate the prior transformation relating each pair of adjacent frames by

$$\hat{\mathbf{t}} = \frac{\mathbf{t}_{1:n}}{n-1}, \quad \hat{A} = (A_{1:n})^{1/(n-1)}.$$

Thus, $\{\boldsymbol{\mu}_n, \Sigma_n\}$ is the result of incrementally applying the transformation $\{\hat{A}, \hat{\mathbf{t}}\}$ to $\{\boldsymbol{\mu}_1, \Sigma_1\}$ $n-1$ times. Figure 2 shows an example linear interpolation of the affine parameters.

There are many other ways to estimate $\hat{\mathbf{a}}$; this method was chosen for its simplicity of implementation. Other linear interpolations exist because there are other ways of parameterizing the Gaussian distribution. For example, we could instead search for the transformation $\hat{\mathbf{a}}$ that contains as little scaling as possible. Or, instead of fitting a line to the object parameters at the start and end of the occlusion event, we could fit a spline. This spline would be influenced by heuristics that estimate the likelihood of each parameterization at each frame in the occlusion event. We would then estimate $\hat{\mathbf{a}}_{t:t+1}$ as the transformation that takes the model along the spline at frame t to the model along the spline at frame $t+1$. We plan on exploring alternatives to our linear interpolation in future work.



Figure 2: An example showing how the mean and covariance of the object (in this case a mouse) on the left is linearly interpolated into the mean and covariance of the object on the right, using our algorithm for linear interpolation. The leftmost ellipse corresponds to $(\boldsymbol{\mu}_1, \Sigma_1)$ and the rightmost ellipse corresponds to $(\boldsymbol{\mu}_n, \Sigma_n)$. The affine prior $\hat{\mathbf{a}}$ transforms any of these ellipses to the ellipse on its right.

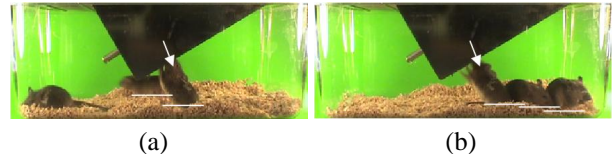


Figure 3: Lowest-pixel depth-ordering heuristic. (a) A frame in which the object with the lowest y -coordinate is the front object. (b) A frame in which the lowest y -coordinate heuristic disagrees with the true depth ordering. The object on the left is the true front object, but the other two objects in the frame have lower extreme y -coordinates.

2.3.3 Depth Ordering Heuristics

Estimating the correspondence between objects at the start and end of an occlusion event relies on two heuristics to compute the depth ordering. The first is that the front object is the object that owns the pixel with the lowest (largest) y -coordinate. This heuristic requires an unoccluded environment in which the floor is visible and the camera is above the floor. It also requires that the layer object model hold, i.e. there is no significant depth variation within a single object. Figure 3 shows an example in which this depth heuristic works and an example in which it doesn't. Because this estimate is susceptible to noise in the layer segmentation, we used the lowest (largest) y -coordinate in the first 3 frames outside the occlusion. The cue computed when comparing depths of a pair of objects is the difference in the largest y -coordinate:

$$s_{low}(m_1, m_2) = (y_{low,1} - y_{low,2}) / \lambda_{low},$$

where λ_{low} is a constant. We set λ_{low} to an estimate of the maximum difference between a pair of lowest pixels.

The second depth ordering heuristic method measures image edge strength around the boundaries of the predicted object positions. For each pair of objects, we determine

which pixels are in (or nearly in) the intersection of the predicted object positions. For each of these pixels, we compute the edge strength using the Sobel edge enhancer, yielding a vector of values \mathbf{e} . For each of these pixels, we also estimate the minimum Euclidean distance between the pixel location and each object m 's boundary, \mathbf{d}_m . We then compute the dot product of these vectors for each object m ,

$$s_{contour}(m_1, m_2) = (\mathbf{e}^\top \mathbf{d}_2 - \mathbf{e}^\top \mathbf{d}_1) / n_{12},$$

n_{12} is the number of pixels in (or nearly in) the intersection.

If the total $s(m_1, m_2) = s_{lowest}(m_1, m_2) + s_{contour}(m_1, m_2)$ is above zero, then object m_1 is declared to be in front of object m_2 .

2.3.4 Layer Segmentation

Given the estimates of the affine motions transforming the layers in the first frame to the layers in the second frame, the pixels in the second frame are segmented into layers.

We assume a static camera, and segment the background layer using learned background and foreground appearance models. A background pixel should be similar to the background appearance, dissimilar to the foreground appearance, and far from the object positions predicted by BraMBLe. We threshold a combination of these properties to segment the background layer.

We apply the predicted affine motions of each layer m , $\{A_m, \mathbf{t}_m\}$, to the object configuration in the first frame, X_1 , to predict the object configuration in the second frame, $\hat{X}_2 = f(X_1, \{A_m, \mathbf{t}_m\})$. As inevitably there will be some error in the affine motion estimations, using just these predictions will cause the object positions to drift from the image data. In addition, we would like the local optical flow of each pixel in the layer to be similar to the motion of the entire layer. We therefore perform a local search around the predicted object configuration, \hat{X}_2 , for a configuration X_2 that is consistent with the image appearance and the local optical flow estimates and near the predicted object configuration.

We now describe the exact implementation of this search. We generate a particle set of object configurations, $\{(X_2^n, 1/N) : n = 1, \dots, N\}$ representing the normal distribution centered on the predicted configuration, \hat{X}_2 . This particle set represents the prior distribution of object configurations. We measure the observation likelihood for each of these configurations,

$$p(Z|X_2^n) = p(Z_{app}|X_2^n)p(Z_{mot}|X_2^n).$$

The appearance observation likelihood is equivalent to the observation likelihood computed by BraMBLe. It enforces that each pixel in an object be similar to the foreground appearance and dissimilar to the background appearance.

The motion observation likelihood has a similar form, enforcing that the local optical flow of each observable pixel in an object be similar to the predicted translation of the entire object, \mathbf{t}_m . Specifically, we assume each pixel's local optical flow is normally distributed around the predicted regional translation of the object. The particle set $\{(X_2^n, p(Z|X_2^n)) : n = 1, \dots, N\}$ represents the posterior distribution $p(X_2|Z)$. We estimate the object configuration in the second frame as the expected value of this posterior distribution.

We use the configuration X_2 to segment the foreground pixels in the second frame. If a foreground pixel belongs to one or more objects defined by X_2 , then it also belongs to the layers representing those objects. If a foreground pixel does not belong to any object, it is assigned to the closest object in terms of Mahalanobis distance.

2.3.5 Reasoning Forward and Backward in Time

We update the prior estimate $\hat{\mathbf{a}}$ as we compute the positions of the objects in a new frame. We would like to perform the affine motion estimation in the pairs of frames that give the most reliable results first. We therefore simultaneously track objects forward in time through the first pair of frames and backwards in time through the last pair of frames, then forward through the second pair of frames and backward through the second-to-last pair of frames, and so on. We do this because the occlusion occurs and finishes gradually, and also because the longer we track overlapping objects, the more our predicted object positions drift from the true positions.

As a concrete example of how this symmetric reasoning can improve accuracy, consider an occlusion which is very difficult to track through in the first half, but easy in the second half. If we only reason forward in time, our position estimates are likely to drift from the true object positions by the middle of the occlusion, and hence for the rest of the occlusion. However, reasoning backwards in time as well, we would successfully track the objects through the second half of the occlusion. As the per-frame motion hint is updated at each iteration, the correct tracking in the second half of the occlusion will help disambiguate the first half of the occlusion.

Thus, in the i^{th} iteration of the algorithm, we first estimate the per-frame motion guess $\hat{\mathbf{a}}$ based on the positions computed by the occlusion reasoning algorithm for frames i and $n - i + 1$, where n is the number of frames in the occlusion. We then track the objects forward in time through the i^{th} pair of frames, and backward in time through the $(n - i + 1)^{\text{th}}$ pair of frames. This involves predicting the "best" forward and backward layer-based affine motion for each layer in frames i and $n - i + 1$, respectively. Then, the layers in frames $i + 1$ and $n - i$ are estimated using the

computed “best” affine motion estimates.

When the two tracks meet in the middle of the occlusion, at frames $n/2$ and $n/2 + 1$, we compute the fraction of points in the intersection of each pair consisting of one object in frame $n/2$ and one object in frame $n/2 + 1$. We match up object identities to maximize the fractions of intersection. If multiple correspondences between objects in frames $n/2$ and $n/2 + 1$ are more or less equally good, we rely on the depth orders to match up the objects.

```

For each frame  $t$ :
1. Estimate the object configuration  $X_t$  using BraMBLe.
2. For each pair of objects  $m_1$  and  $m_2$ ,
   If  $m_1$  and  $m_2$  are separated, check for the
   start of an occlusion between  $m_1$  and  $m_2$ .
   Otherwise, check for the end of the
   occlusion between  $m_1$  and  $m_2$ .
3. If occlusion end detected, perform occlusion
   reasoning.

```

Tracking Algorithm

```

For  $i = 1$  to  $n/2$ :
1. Compute the guesses  $\hat{a}$  based on the object configurations  $X_i$  and  $X_{n-i+1}$ .
2. Compute the “best” forward affine motion estimates  $\mathbf{a}_{i,i+1}$  for each layer in frame  $i$ .
3. Segment the pixels in frame  $i+1$  into motion layers.
4. Compute the “best” backward affine motion estimates  $\mathbf{a}_{n-i+1,n-i}$  for each layer in frame  $n-i+1$ .
5. Segment the pixels in frame  $n-i$  into motion layers.

```

Occlusion Reasoning

Figure 4: Algorithm Summary

3. Experiments

We tested our algorithm on video of three indistinguishable mice exploring a cage, available at <http://vision.anonymous.edu>. We tested our algorithm on the first 1800 frames of data collected. The first 200 frames were used for background initialization and tracking was started at frame 236, in which there is no occlusion. Each frame initially had size 240×320 , but we cropped the image at the boundaries of the cage, resulting in frames of size 121×313 . We first describe the practical details of our implementation.

3.1. Implementation Details

Our implementation of BraMBLe is slightly modified relative to [6]. First, each object is modeled by an ellipse parameterized by its center, orientation, and minor and major axes lengths. We use this parameterization throughout the algorithm. Second, we have a “closed-world” environment, so we do not let the number of objects change (though an object can be reinitialized). Third, we restrict our ellipses to be of reasonable sizes. The parameters were determined by making guesses at the amount a mouse moves

between a pair of frames and trying to make the variance in the response likelihoods for the background similar to that of the foreground. In addition, the variance added to the background model and the size of the uniform distributions mixed with the background and foreground appearance models were necessary to avoid numerical overflow/underflow.

The parameter settings for our occlusion reasoning, occlusion detection, and depth ordering algorithms were set using intuition about the problem. However, as described in [2], the algorithm is not particularly sensitive to most of these parameters. Parameter settings and more details can be found in our technical report.

3.2. Results

Our results are summarized in Figures 5 and 6. For purposes of visualization, we show in Figure 5(a) a single scanline of the image (row 93 of the cropped image) at every frame of the video sequence. Row 93 was chosen as it passes through the mice most of the time. The horizontal axis of this image is time and the vertical axis is the x -axis of an original frame. Each dark path from left to right in this (t, x) image is the path a single mouse takes through the sequence; notice there are three paths.

Let us call the mouse that starts at the top of this (t, x) image mouse 1, the mouse that starts in the middle mouse 2, and the mouse that starts at the bottom mouse 3. In this sequence, the mice begin unoccluded, then in frame 20 mouse 3 goes behind mouse 2, turns around, and comes out from behind mouse 2 in frame 92 on the same side on which it entered. Our algorithm correctly tracks object identities through this occlusion.

In frame 204, mouse 1 goes partly behind mouse 2. In frame 346, mouse 3 also goes partly behind mouse 2. It walks behind both mice 1 and 2, and leaves the occlusion on the other side. Mouse 1 then leaves the occlusion on the same side on which it entered in frame 606. During this occlusion event, our algorithm swaps the identities of mice 1 and 3. Because mouse 3 is one of two back objects, the affine motion guess computed reflects the motion of both the mice. In addition, the occlusion is extremely long (over 400 frames long), thus the per-frame motion guess is small. When mouse 3 is totally occluded by the other two mice, motion estimation relies only on the motion guess. As the motion guess is small, the position estimate gets stuck behind mouse 2, until the gap between the true and estimated positions is too large to recover from. We discuss in Section 4 future work that may allow this mouse to be tracked. Notice the discontinuity at frame 410 in the green path in Figure 5.

In frame 699, mouse 3 passes in front of mouse 1, then goes behind mouse 2, and comes out on the other side of mouse 2 in frame 800. In frame 876, mouse 1 passes behind

Start	End	Correct	Swaps
20	92	.6	.4
204	606	0	2.2
699	800	.4	.6
945	1101	0	1
1103	1347	.2	1.4

Table 1: Results of running BraMBLe on the occlusions. The table shows the start frame of the occlusion, the end frame of the occlusion, the fraction of times identities were correctly tracked, and the average number of times a pair of identities swapped across all trials. Results are for 5 trials of BraMBLe.

mouse 2 and comes out on the other side in frame 930. In frame 945, mouse 1 goes in front of mouse 3, walks to the front of the cage, turns around, and leaves the occlusion with mouse 3 on the same side on which it entered. Our algorithm is successful in both of these occlusions.

In frame 1103, mouse 1 goes behind mouse 2 and stops. Simultaneously, mouse 2 sits down and turns around while mouse 3 enters the occlusion with the other two mice. Mouse 3 passes behind both mice and leaves the occlusion on the other side. Mouse 2 leaves the occlusion on the side on which mouse 1 entered. We swap the identities of mice 1 and 2. This is to be expected, as our depth estimate at the end of the occlusion is wrong. This is because the end of the occlusion is not detected until frame 1304, which is just after the true end of the occlusion. During the gap between the true end and the detected end, mice 1 and 2 switch depth order, as shown in Figure 6. Note that again, when the identities are switched, there is a discontinuity in the path labels.

To illustrate the need for proper occlusion reasoning, we tested BraMBLe on each of these occlusions. We ran BraMBLe for five trials on each occlusion whose start and end points were manually labeled, as BraMBLe computes very different results for different random seeds. The results are summarized in Table 1. In general, BraMBLe performed poorly at tracking identities during the occlusions. This is because, given only the information used by BraMBLe, the occlusions are ambiguous. In addition, the constant velocity assumption does not always hold for the mice in this sequence.

4. Discussion and Future Work

We have presented an algorithm to track multiple indistinguishable objects through severe occlusions. The algorithm breaks the tracking task into three subproblems: separated object tracking, occlusion detection, and occlusion reasoning. We chose the BraMBLe algorithm to track separated mice and detect occlusions because BraMBLe uses the correct model for occluded pixels. The main contribution of this paper is an algorithm for occlusion reasoning. We be-

lieve that this algorithm can be added to virtually any tracking algorithm that meets the requirements discussed in Section 2.1.

This algorithm uses a depth ordering heuristic to match up the front object at the start of an occlusion with the front object at the end of the occlusion. This correspondence is used as a hint that is combined with the optical flow cue. A single frame of an occlusion event out of context can have multiple fits that seem equally good. The “premonition” of the final position of the object gives our algorithm a way of deciding between these equally good fits.

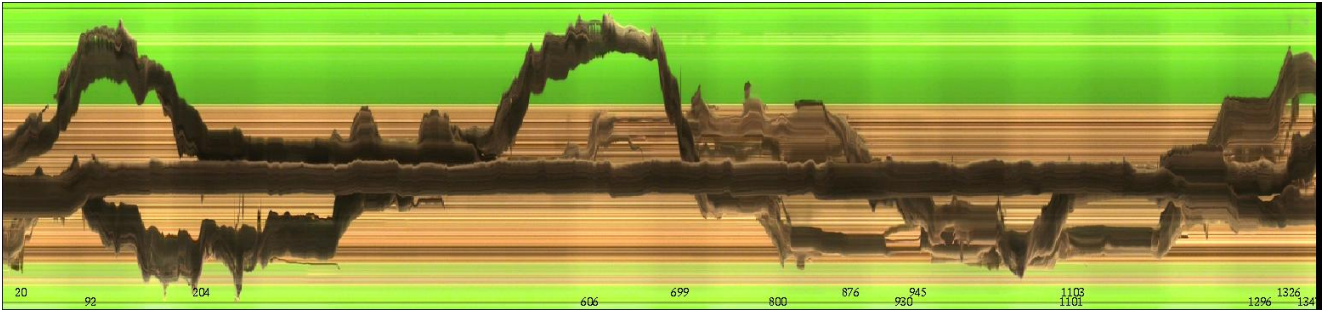
Our algorithm simultaneously reasons forward and backward in time, continually updating the hint. Reasoning forward and backward in time, as opposed to just forward in time, improves the accuracy of the algorithm.

We tested our algorithm on a challenging video sequence of three mice in a cage. The mice are indistinguishable and have erratic motions. We incurred swapped identities twice in this sequence. The first swap, as discussed in section 3, was because the occlusion was long and the object motion did not fit the linear interpolation. We plan on trying nonlinear interpolation schemes, such as a spline interpolation, to solve this problem. In addition, we are working on a method for recursively applying our tracking algorithm to track just the back mice. The difficulty in this task is that occlusion detection and depth-ordering is much more difficult when the objects are occluded.

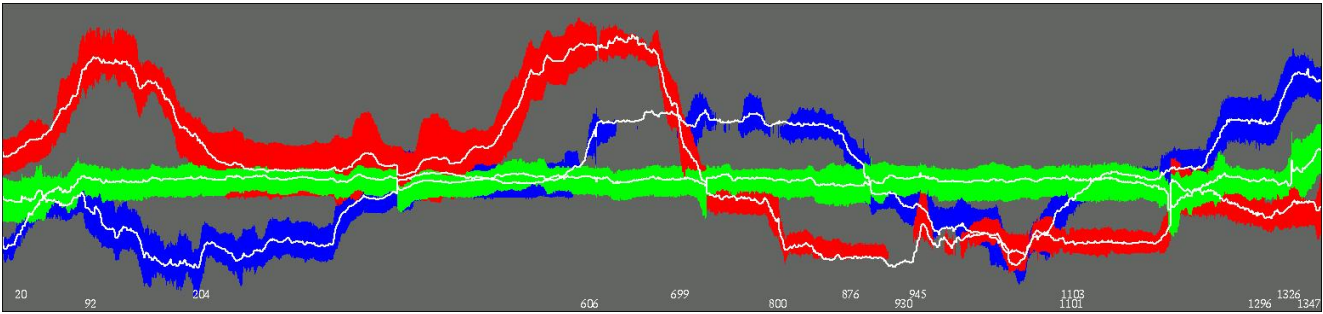
In conclusion, the major contribution of this work is a method for tracking indistinguishable, featureless targets through occlusion events by combining multiple cues in a noncausal fashion throughout the duration of each occlusion event.

References

- [1] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. In *Pattern Analysis and Machine Intelligence*, volume 25 (5), 2003.
- [2] Blank for anonymous review.
- [3] J. Gårding. *Shape from surface markings*. PhD thesis, Royal Institute of Technology, Stockholm, 1991.
- [4] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer Verlag, Basel, 2001.
- [5] M. Irani and P. Anandan. All about direct methods. In *Vision Algorithms: Theory and Practice*. Springer-Verlag, 1999.
- [6] M. Isard and J. MacCormick. BraMBLe: A Bayesian multiple-blob tracker. In *ICCV*, 2001.
- [7] J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. *IJCV*, 39(1):57–71, 2000.
- [8] H. Tao, H. Sawhney, and R. Kumar. A sampling algorithm for tracking multiple objects. In *Workshop on Vision Algorithms*, pages 53–68, 1999.



(a) (t, x) raw image data (320×1365 pixels): a single scanline of the image at every frame.



(b) (t, x) predicted image (320×1365 pixels): membership of points in a scanline of the image at every frame.

Figure 5: **Tracking results** (t, x) plot of results. The horizontal axis in these images is time and the vertical axis is the x -axis of the original frame. Each column corresponds to the same scanline of a different frame. If a path disappears, it is because the chosen scanline does not pass through the mouse.

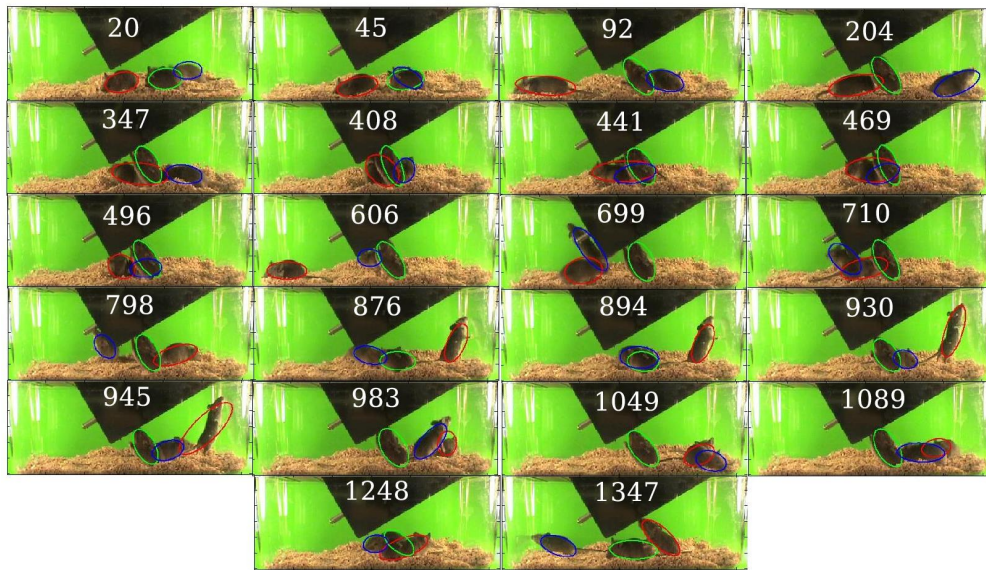


Figure 6: Example frames illustrating our tracking results. The ellipses are the object shapes estimated by our algorithm. Frame numbers are printed in the images.