Three Brown Mice: See How They Run

Kristin Branson, Vincent Rabaud, and Serge Belongie
Department of Computer Science and Engineering
U.C. San Diego
La Jolla, CA 92093
http://vision.ucsd.edu

Abstract

We address the problem of tracking multiple, identical, nonrigid moving targets through occlusion for purposes of rodent surveillance from a side view. Automated behavior analysis of individual mice promises to improve animal care and data collection in medical research. In our experiments, we consider the case of three brown mice that repeatedly occlude one another and have no stable trackable features. Our proposed algorithm computes and incorporates a hint of the future location of the target into layer-based affine optical flow estimation. The hint is based on the estimated correspondences between mice in different frames derived from a depth ordering heuristic. Our approach is simple, efficient, and does not require a manually constructed mouse template. We demonstrate encouraging results on a challenging test sequence containing multiple instances of severe occlusion.

1. Introduction

In this paper we consider the problem of multiple object tracking in the case where the objects are indistinguishable and prone to occluding one another. Recently a number of works have appeared that address the problem of multiple object (or blob) tracking [5, 1, 10]. Many of these approaches leverage object-specific appearance models such as color histograms [1]. Most closely related to our problem setting is that for which the BraMBLe algorithm was designed [5]; this reference also provides a thorough review of relevant visual tracking methods. In this work, the authors use a particle filter to track multiple blobs (viz. people) from a ceiling-mounted hallway camera. The principal failure mode of their system is that of blob labels (i.e. identities) getting switched when one object passes in front of another. The authors suggest that the use of individual foreground models for each object could be used to solve this problem. In our setting, however, the objects we wish to track are mice, and they all have the same appearance; see Figure 1.

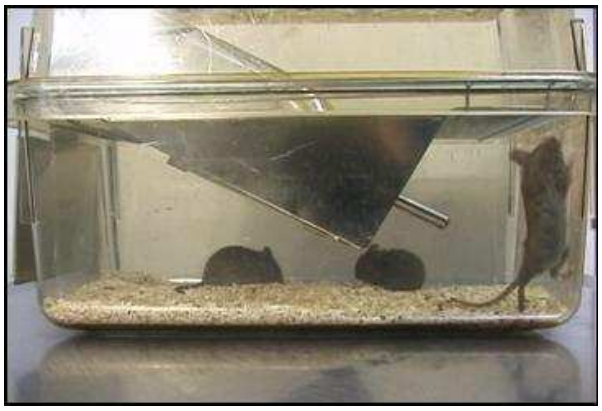


Figure 1: Still frame captured from video sequence of three mice (240×360 pixels). The metal container at the top of the cage holds food pellets and a water bottle. It also prevents the use of an overhead mounted camera. The bedding on the floor of the cage is the only dynamic part of the background, other than reflections.

Aside from the challenge of tracking identical targets through occlusion, our problem setting presents a number of other difficulties. The objects we wish to track have few if any trackable features (in the sense of [9]) that last for more than a few frames. Additionally, the motion of the objects is relatively erratic compared to a car driving past an occluding signpost, for example. On the other hand, we benefit from a number of simplifying assumptions, e.g. that the number of objects does not change, the illumination is relatively constant, and the camera is stationary.

Because of these simplifying assumptions, the subproblems of foreground/background classification and tracking for separated mice (mice that are neither occluded or occluding) are adequately addressed by many existing algorithms. The subproblem remaining is to track the mouse identities while they are occluding one

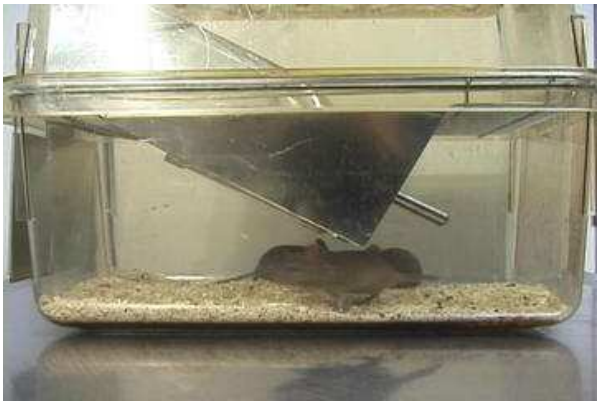


Figure 2: Example frame in which distinguishing the boundaries of both the back mice as well as the front mouse is difficult.

another. Given the foreground/background labeling, this subproblem reduces to assigning membership of each foreground pixel to the mouse identities. The difficulty of this subproblem is illustrated in Figure 2. In this example, the back mice are almost completely occluded and it is difficult to determine the boundary of the front mouse.

As segmenting the individual mice is more difficult when a frame is viewed out of context of its neighboring frames, we incorporate a cue from the surrounding sequence of frames. Using a depth ordering heuristic, we associate the front mouse at the start of an occlusion event with the front mouse at the end of an occlusion event. This correspondence is valid because objects cannot pass through each other; thus while one mouse occludes another, their depth ordering cannot change, as observed in [6]. We predict mouse identity labels for each frame sequentially. However, when labeling a frame during an occlusion event, we incorporate the hint of the future location of the mice in addition to the predicted labels of the previous frames.

Because this correspondence guess is necessary in our occlusion reasoning, images can be processed at frame-rate, but tracking results for occluded or occluding mice are delayed until the end of the occlusion event.

Besides being a uniquely challenging testbed for tracking algorithms, mouse tracking technology promises to be a valuable tool for medical research. A single vivarium can contain thousands of cages of mice, making close monitoring of individual mice impossible. Automated behavior analysis of individual mice would allow for improved animal care and more detailed and exact data collection. Improved animal care results from early detection of abnormal behavior. Exact and detailed behavior analysis will improve the efficiency of medical experiments. An algorithm that tracks individual mice is a necessity for automated behavior analysis.

Video surveillance of mice has the important characteristic of being nonintrusive; no modification to the environment is necessary. It is now feasible because of the recent availability of low-cost video cameras. Because of the huge number of medical experiments conducted on caged mice, this feasibility has led to a surprising amount of research on this seemingly obscure problem [7]. To our knowledge, all current approaches (for example, [11]) require overhead mounted cameras. This simplifies the problem because the amount of occlusion is reduced. However, this kind of surveillance requires a specially designed cage, since in a standard mouse cage the overhead view is obstructed by the feeder and cage top (see Figure 1). In this paper, we propose a nonintrusive approach that can work with any reasonable cage design because it uses a side view.

The organization of this paper is as follows. We describe our proposed approach in Section 2. In Section 3 we provide experimental results on real world footage of caged mice. We conclude and discuss future directions in Section 4.

2. Our Approach

Our approach breaks the tracking task into the subproblems of background/foreground classification, tracking separated mice and tracking through occlusion. First, the pixels in each frame are classified as either foreground or background. Next, the membership of the foreground pixels in every frame is assigned using a simple tracking algorithm without occlusion reasoning. The mouse models computed in this step are used to detect the starts and ends of occlusion events. At the end of an occlusion event, the membership of the foreground pixels in the frames of the occlusion event are reassigned using our occlusion reasoning algorithm. The modules used to solve each of these subproblems are described below.

2.1 Background/Foreground Labeling

Next, we describe the simple background/foreground classification we used. This algorithm was sub-optimal, but did not corrupt our results. We plan on experimenting with more sophisticated algorithms, as described in Section 4.

To classify pixels in every frame as background or foreground, we model the background using a modified temporal median. The absolute difference of the

current frame and the current background estimate is thresholded. Background pixels are those below the threshold and foreground pixels are those above the threshold.

Standard background estimation using a temporal median estimates the background intensity of each pixel as the median of the previous n_B intensities of that pixel. This technique fails when more of the previous n_B intensities of a pixel correspond to foreground than background. As it often occurs that a mouse (e.g. a sleeping mouse) is still for many frames at a time, we include an additional component depending on the color of the pixel. Each pixel is classified as either mouse color or not mouse color. The intensity of each pixel classified as mouse color is only added to the pixel's intensity history if that pixel has always been classified as mouse color.

To classify the pixels in a frame as mouse color or not, the pixels are segmented into a set number of clusters (we used 3) based on color using k-means. In order to determine which color cluster(s) correspond to mouse color (we assumed only one cluster was mouse color), we use the previous background estimate to classify pixels as background and foreground. The mouse color cluster(s) are the mode color cluster(s) of the foreground pixels.

2.2 Separated Target Tracking

In all frames, whether or not there is occlusion, the distribution of the pixel locations of each of the k mice identities is modeled as a bivariate Gaussian. In a frame in which the mice are not occluding each other, the problem of assigning mouse membership to each of the foreground pixels is that of fitting a mixture of k Gaussians to the locations of the foreground pixels. We use the EM algorithm to estimate the mean vectors and (full) covariance matrices of the Gaussian mixture model (GMM) [3]. As the inter-frame motion is small, only a few iterations of EM are necessary when we use the parameters of the GMM at frame $t-1$ to initialize the EM fit at frame t.

2.3 Detection of Occlusion Events

The goal of this module is to determine the starts and ends of occlusion events, as well as which mice and foreground pixels are involved. Occlusion events are detected using the GMM parameters computed using EM.

The Fisher distance in the x direction between each pair of target distributions is thresholded to determine when one mouse of the pair is occluding the other. We use only the x distance because all mice are resting on the floor and the x location estimates of our GMM are more stable. The Fisher distance in x between a pair of distributions is defined as

$$J_F[\mu_{x1}, \sigma_{x1}^2, \mu_{x2}, \sigma_{x2}^2] = \frac{(\mu_{x1} - \mu_{x2})^2}{(\sigma_{x1}^2 + \sigma_{x2}^2)/2},$$

where μ_{x1} and μ_{x2} are the x-coordinates of the distributions' means and σ_{x1}^2 and σ_{x2}^2 are the variance in x of the distributions [3]. Because the units of σ_x and μ_x are the same, the Fisher distance is unitless and a constant threshold (we chose 8.0) is used.

2.4 Tracking through Occlusion

When an occlusion event is detected, the membership of each foreground pixel in the occlusion must be assigned. Given an estimate of the pixels belonging to each mouse in frame t, we compute the "best" affine transformation describing the motion of that mouse from frame t to $t+1$. Given these affine motion estimates, the pixels belonging to each mouse identity in frame $t+1$ are estimated.

In Section 2.4.1 and 2.4.2, we will describe our criterion for the "best" affine motion and how it is optimized. In Section 2.4.3, we will describe how the pixel memberships are estimated.

2.4.1 Optical Flow Computation

Our algorithm for tracking through occlusion is based on optical flow estimation using multiple affine models. Consider the set of pixels belonging to one mouse (e.g. in Figure 1). 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, $I(x, y, t)$ 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 [4], 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 mouse, the vectors $\mathbf{z}$ and $\mathbf{a}$ are defined as

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

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

Because of the high amount of occlusion and the lack of features on the targets, the optical flow cue alone is not enough to get an accurate motion estimate. We thus add a hint of the future mouse 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 2.4.2. We use the term "prior" because of the close relation of this regularization term to an assumed prior distribution on $\mathbf{a}$ [3]. 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. We use $\lambda = 0.0001$. 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:

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

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

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

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 the mean and variance of the location of the mouse correspond to all pixel locations, occluded and unoccluded. It is only safe to assume that the affine transformation for the visible part of the mouse equals the affine transformation for the entire mouse if a significant portion of the mouse 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 mouse is occluded, and rely only on the affine motion prior $\hat{\mathbf{a}}$.

2.4.2 Prior Estimation

Next, we discuss the choice of $\hat{ba}$ for each mouse in each frame of the occlusion. As mentioned before, $\hat{\mathbf{a}}$ can be interpreted as the mean of the prior distribution on $\mathbf{a}$. To estimate $\hat{\mathbf{a}}$, we use only the depth ordering cue, though other cues could be incorporated. The depth ordering cue is an estimate of which blob is in front 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 mouse, we reason that during the occlusion event, the succession of frame to frame motions must transform the initial front mouse to the final front mouse. We cannot assume that the back mice do not change depth ordering with respect to each other during the occlusion. Instead, we assume that the blob of all the back mice at the start of the occlusion corresponds to the blob of all the back mice at the end of the occlusion. We set the prior estimates for each of the back mice to be all the same.

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,

$$A = \begin{pmatrix} a_2 & a_3 \\ a_5 & a_6 \end{pmatrix}, \qquad \mathbf{t} = (a_1, a_4)^\top$$

If the displacement (u,v) is computed in the coordinate system centered on $\boldsymbol{\mu}$ and the pixel locations $\mathbf{p}$ belonging to a mouse follow the normal distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$, then the transformed locations $\mathbf{p}' = \mathbf{p} + (u,v)^\top$ follow the normal distribution $\mathcal{N}(\boldsymbol{\mu}', \Sigma')$, where

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

We first compute the transformation $(A_{1:n}, \mathbf{t}_{1:n})$ that transforms the mouse in the first frame of the occlusion event, described by $\mathbf{p}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)$, to the mouse in the last frame of the occlusion event, described by $\mathbf{p}_n \sim \mathcal{N}(\boldsymbol{\mu}_n, \Sigma_n)$, where n is the number of frames in the occlusion event. Any pair in the family

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

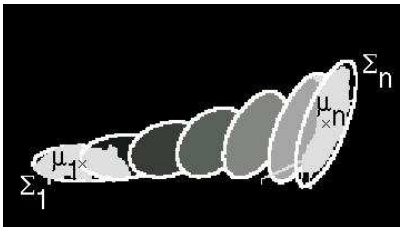


Figure 3: An example showing how the mean and covariance of the mouse on the left is linearly interpolated into the mean and covariance of the mouse on the right, using our algorithm for linear interpolation. The leftmost ellipse corresponds to $(\boldsymbol{\mu}_1, \Sigma_1)$ and the rightmost ellipse corresponds to $(\boldsymbol{\mu}_n, \Sigma_n)$. The affine prior $\hat{\mathbf{a}}$ transforms any of these ellipses to the ellipse on its right.

where O is an arbitrary orthogonal matrix will perform the desired transformation [2]. 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

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

Thus, $\mathcal{N}(\boldsymbol{\mu}_n, \Sigma_n)$ is the result of incrementally applying the transformation $(\hat{A}, \hat{\mathbf{t}})$ to $\mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)$ $n-1$ times. Figure 3 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. Instead of fitting a line to the mouse 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.

To correspondence between mice at the start and end of an occlusion event relies on a very simple heuristic to compute the depth ordering. The front mouse is the mouse that owns the pixel with the lowest (largest) y-coordinate. This is true in any unoccluded environment in which the floor is visible and the camera is above the floor. This depth cue is evident in the example in Figure 1. Because this estimate relies heavily on the noisy foreground classification, we used the lowest (largest) y-coordinate in the past 5 frames as a depth estimate.

2.4.3 Pixel Membership Estimation

Given the estimates of the affine motions transforming the mice at frame t to the mice at frame $t+1$, the foreground pixels belonging to each mouse in frame $t+1$ are estimated. The pixels belonging to one mouse should be similar in both motion and location, as motivated in [12]. In order to incorporate both motion and location, we assign membership based on the weighted sum of proximity and motion similarity.

We estimate the mean and variance of the location of a mouse at frame $t+1$ by applying the computed affine transformation to the estimated mean and variance of the location of the mouse at frame t. The proximity criterion for a pixel at location $\mathbf{p}$ is

$$J_l[\mathbf{p}] = (\mathbf{p} - \boldsymbol{\mu}_{t+1})^\top \Sigma_{t+1}^{-1} (\mathbf{p} - \boldsymbol{\mu}_{t+1}).$$

We also compute the local optical flow for each foreground pixel. For this, we use Lucas-Kanade with a Gaussian window with standard deviation 2.0. Note that this re-uses the spatiotemporal derivatives used in the affine flow estimation. The optical flow of each pixel in a mouse should be similar to the regional optical flow of the entire mouse. The motion similarity term for a pixel $\mathbf{p}$ with motion $(u_{local}, v_{local})^\top$ is

$$J_m[\mathbf{p}] = \lambda_{local}[(u_{local} - a_1)^2 + (v_{local} - a_4)^2].$$

Our total cost function is therefore

$$J[\mathbf{p}] = J_l[\mathbf{p}] + J_m[\mathbf{p}].$$

Each relevant foreground pixel is assigned to the mouse with the lowest summed location and motion similarity terms.

2.4.4 Tracking the Back Mice

To track the back mice, we reapply the algorithm to just the back mice. Because the back mice might be occluded by the front mouse at the start or end of their occlusions, the depth ordering heuristic is much less reliable. We do not use this heuristic if any of the back mice in the occlusion are significantly occluded by the front mouse. Instead, we use $\hat{\mathbf{a}} = \mathbf{0}$, thus the regularization term shrinks the optical flow estimates.

2.5 Parameter Sensitivity and Computational Considerations

We have mentioned the parameter settings we used in our experiments throughout this section. These pa-

rameters weight the different terms in our optimization. The parameter λ, the weight of the prior term in the flow estimatio, was set to 10^{-4}. However, the algorithm is not particularly sensitive to this parameter. Values in the range 10^{-5} to 5×10^{-4} produced similar results. This insensitivity and the small size of effective λ settings is due to the ability of the affine flow estimation stage to rely solely on the prior estimate when too much of the mouse is occluded. In future work, we will experiment with a dynamically computed λ based on an estimate of the reliability of the optical flow cue.

Parameters λ_u and λ_v are the weight of the motion criterion in the mask estimation. These were both set to 1.0, but values between 0.25 and 2 produced similar results. As λ_u and λ_v correspond to the inverse of the variance of u_{local} and v_{local}, respectively, dynamic estimation may also work for these parameters.

The matrix Σ_a was set to $\text{diag}\{1, 0.1, 0.1, 1, 0.1, 0.1\}$. As Σ_a corresponds to the relative variance of $\mathbf{a}$, we plan to fit Σ_a from actual data.

The running time of our occlusion reasoning module is linear in the total number of foreground pixels in the frames of the occlusion event. While our current implementation is in Matlab™, we believe that a more efficient implementation of this module will run in real time. Currently, the occlusion reasoning takes about 0.642 seconds per frame in our experiments with three mice on a 2.4 MHz Pentium 4. Approximately 0.172 seconds of this time is involved in affine flow estimation, 0.0529 seconds of this time is involved in mask estimation, and the rest of the time is overhead from parameter passing in functions.

3. Experiments

We report the initial success of our algorithm in tracking three mice in a cage.

3.1 Experimental Setup

We tested our algorithm on a 1000-frame video sequence (available at `http://vision.ucsd.edu`) taken from the side of a Static Micro-Isolator™cage containing three adolescent mice. The first 200 frames were used for background initialization and tracking was started at frame 225, in which there was no occlusion. Figure 1 shows an example frame from this sequence. The cage is made of a translucent plastic and is approximately the size of a shoe box. At the top of the cage is a metal container with food pellets in one half and a water bottle in the other half. The camera was positioned slightly above the level of the mouse floor. Because of reflections in the table and the top of the cage, we cropped each initially 240×360 pixel frame at the top grate (row 60) and the bottom of the cage (row 193), resulting in 134×360 pixel frames. The video was recorded at 30 frames/second and compressed into Windows Media Video (wmv) format.

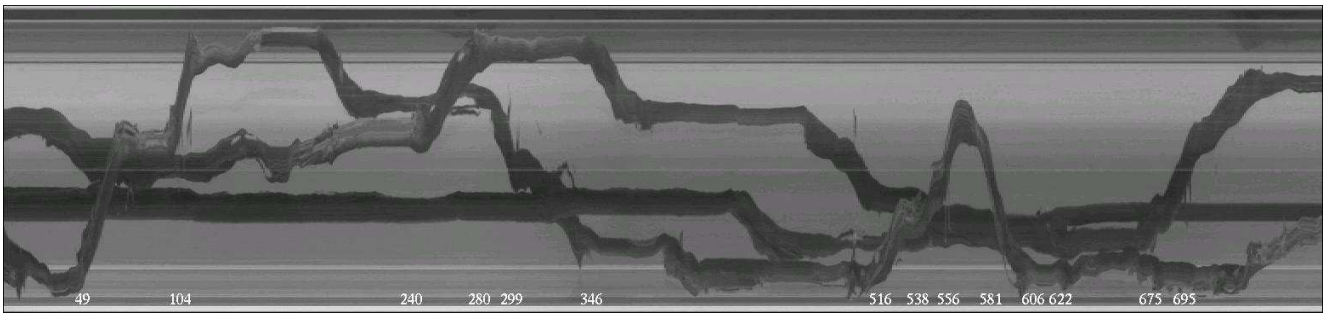
3.2 Results

Our results are summarized in Figures 4 and 5. For purposes of visualization, we show in Figure 4(a) a single scanline of the image (row 96 of the cropped image) at every frame of the video sequence. Row 96 was chosen as it passes through the middle of the mice. The x-axis of this image is time and the y-axis is actually 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.

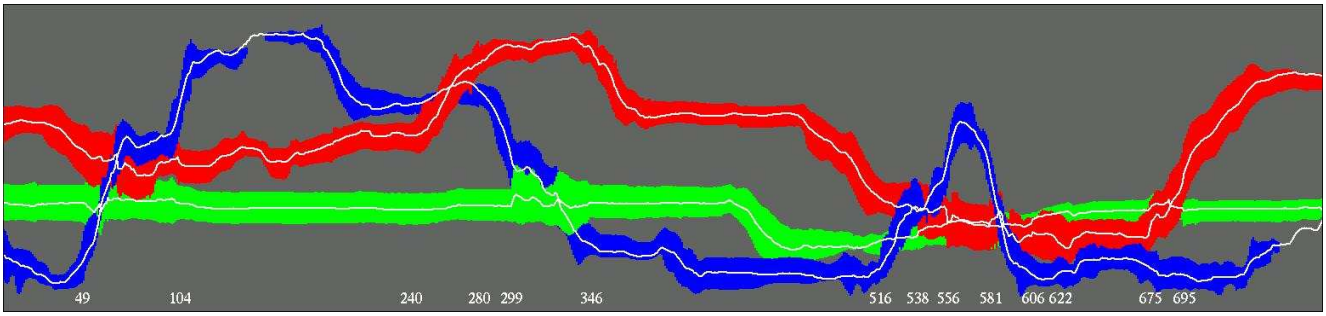
Let us call the mouse that starts at the top of this (t, x) image mouse 1, the mouse that starts in the middle mouse 2, and the mouse that starts at the bottom mouse 3. In this sequence, the mice begin unoccluded, then in frame 49 they all move together and mouse 3 passes in front of the other two. In frame 240, mouse 1 passes in front of mouse 3. In frame 299, mouse 3 passes behind mouse 2. In frame 516, all the mice again come together and mouse 3 passes in front, then turns around and again passes in front of them in frame 581. Meanwhile, in frame 538 mouse 1 passes in front of mouse 2. Finally, in frame 675, mouse 1 passes in front of mouse 2.

In Figure 4(b), we show the labels estimated by our algorithm in this (t, x) format. At each frame, each point along the scanline (row 96) within two standard deviations of a mouse is plotted with a color corresponding to that mouse's label. Thus, each path of one color is the estimated path of a mouse. There are two breaks in the path of mouse 3; these occur because two standard deviations of the estimated Gaussian does not intersect the chosen scanline (because of errors in the foreground classification). We also plot the x-component of the centers of the estimated Gaussians in white. For points predicted to belong to multiple mice, we plot the color of the mouse predicted to be in front.

In this 776 frame sequence, the identities of the mice are never switched. In fact, the estimated mouse paths match the actual mouse paths very closely. In Figure 5, we show some example image frames and the mouse parameters estimated by our algorithm for two occlusion events.

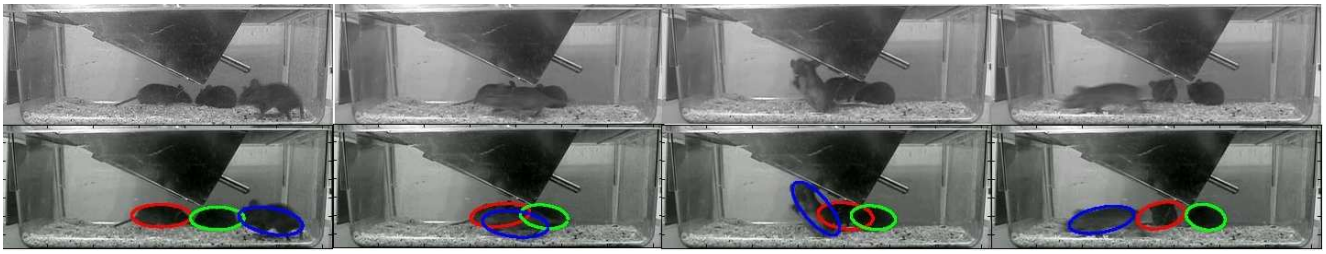


(a) (t, x) raw image data (360×776 pixels): a single scanline of the image at every frame.

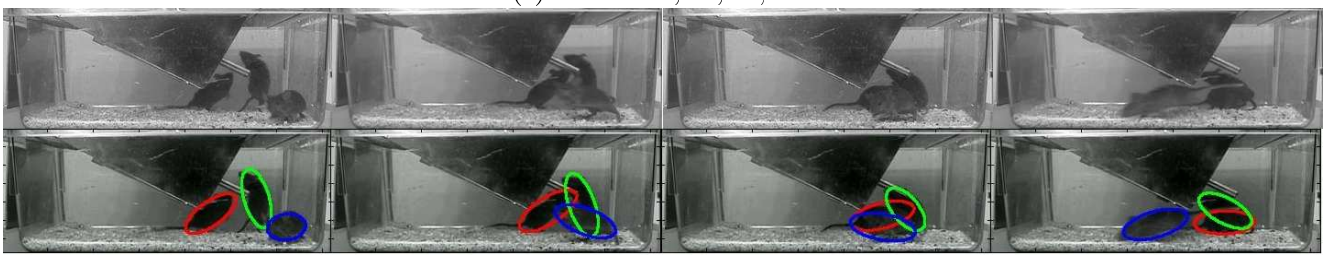


(b) (t, x) predicted image (360×776 pixels): membership of points in a scanline of the image at every frame.

Figure 4: **Tracking results** (t, x) plot of results. The x-axis in these images is time and the y-axis is the x-axis of the original frame. Each column corresponds to the same scanline of a different frame.



(a) Frames 49, 64, 80, 104



(b) Frames 516, 525, 539, 556.

Figure 5: Example frames showing the raw image frames from an occlusion event in the top row and the Gaussian parameters estimated by our algorithm. The ellipses correspond to 2 standard deviations of the Gaussians.

4 Discussion and Conclusion

We have presented a collection of modules that combine many cues to track identical, non-rigid, featureless objects through severe occlusions. The novel module is the occlusion tracking module. This module uses a depth ordering heuristic to match up the front mouse at the start of an occlusion with the front mouse at the end of the occlusion. This correspondence is used as a hint that is combined with the optical flow cue. Because of the nature of our targets and the high amount of occlusion, a single frame out of context can have multiple fits that seem equally good. The "premonition" of the final location of the mouse gives our algorithm a way of deciding between these equally good fits. This module is thus a step in the direction of an algorithm that reasons both forward and backward in time.

Mouse identity was tracked without error in all 776 frames, despite noise in the foreground classification, occlusion detection, and depth estimation. We have thus presented an approach for assigning unique identity labels to mice through occlusions that can work with with other suboptimal modules. These imperfections do not corrupt the occlusion reasoning module because the true depth ordering of the mice does not change rapidly and enough of the true foreground pixels are classified as foreground to segment the motion.

In addition, as our algorithm combines numerous cues, one module's failure does not cause the entire algorithm fail. For example, in one case the foreground classification was poor (it missed the feet of a mouse), which led to an incorrect estimate of the depth ordering in an occlusion event. As this occlusion was not complete, there were sufficient intensity cues to overcome the incorrect depth correspondence. The algorithm succeeded despite the incorrect depth estimate.

Our algorithm did not fail on the sequence we tested because all the modules never failed simultaneously. The primary risk of failure arises from errors in the depth estimation. To prevent this, we are exploring other algorithms to which we can add our occlusion reasoning step. We are considering other algorithms that are more robust than our choice of using independent foreground classification and tracking modules. Specifically, we would like to try adding our occlusion reasoning module to the BraMBLe tracker. We would also like to develop a more robust depth estimation heuristic that does not rely so heavily on foreground classification. This can include a smoothed estimate that uses the assumption that depth changes slowly as well as occlusion junction detection [8] when our depth ordering heuristic fails (e.g. for back mice).

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.

Acknowledgments

The authors wish to thank Sameer Agarwal, Ben Ochoa, Eric Wiewiora, and Josh Wills for valuable discussions. We would also like to thank Keith Jenne, Phil Richter, Geert Schmid-Schoenbein, and John Wesson for providing the video data and valuable suggestions. This work was funded by Cal-$(IT)^2$ – the California Institute for Telecommunications and Information Technology – under the Smart Vivarium project.

References

[1] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. In *Pattern Analysis and Machine Intelligence*, volume 25 (5), 2003.

[2] Jonas Gårding. *Shape from surface markings*. PhD thesis, Royal Institute of Technology, Stockholm, 1991.

[3] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer Verlag, Basel, 2001.

[4] M. Irani and P. Anandan. All about direct methods. In *Vision Algorithms: Theory and Practice*. Springer-Verlag, 1999.

[5] M. Isard and J. MacCormick. BraMBLe: A Bayesian multiple-blob tracker. In *Internation Conference on Computer Vision*, 2001.

[6] John MacCormick and Andrew Blake. A probabilistic exclusion principle for tracking multiple objects. *International Journal of Computer Vision*, 39(1):57–71, 2000.

[7] *Measuring Behavior: International Conference on Methods and Techniques in Behavioral Research*, 1996–2002.

[8] Sourabh A. Niyogi. Detecting kinetic occlusion. In *ICCV*, pages 1044–1049, 1995.

[9] Jianbo Shi and Carlo Tomasi. Good features to track. In *Computer Vision and Pattern Recognition*, Seattle, June 1994.

[10] Hai Tao, Harpreet S. Sawhney, and Rakesh Kumar. A sampling algorithm for tracking multiple objects. In *Workshop on Vision Algorithms*, pages 53–68, 1999.

[11] C. J. Twining, C. J. Taylor, and P. Courtney. Robust tracking and posture description for laboratory rodents using active shape models. In *Behavior Research Methods, Instruments and Computers, Measuring Behavior Special Issue*, 2001.

[12] Yair Weiss and Edward H. Adelson. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *Computer Vision and Pattern Recognition*, pages 321–326, 1996.