Synthetic Aperture Tracking: Tracking through Occlusions

Neel Joshi[†] Shai Avidan[*] Wojciech Matusik[‡] David J. Kriegman[†]
[†]University of California, San Diego [*]MERL [‡]Adobe Systems Inc.

Abstract

Occlusion is a significant challenge for many tracking algorithms. Most current methods can track through transient occlusion, but cannot handle significant extended occlusion when the object's trajectory may change significantly. We present a method to track a 3D object through significant occlusion using multiple nearby cameras (e.g., a camera array). When an occluder and object are at different depths, different parts of the object are visible or occluded in each view due to parallax. By aggregating across these views, the method can track even when any individual camera observes very little of the target object. Implementation-wise, the methods are straightforward and build upon established single-camera algorithms. They do not require explicit modeling or reconstruction of the scene and enable tracking in complex, dynamic scenes with moving cameras. Analysis of accuracy and robustness shows that these methods are successful when upwards of 70% of the object is occluded in every camera view. To the best of our knowledge, this system is the first capable of tracking in the presence of such significant occlusion.

1. Introduction

Tracking an object through a video sequence is a critical step in many computer vision applications. Tracking works by finding a region in each video frame that matches the appearance of a given target object; however, in the presence of occlusion this matching process is severely hampered and often fails. In the case of single-camera tracking, if there is significant occlusion and matching fails, the only recourse is to detect the occlusion [1, 4] and potentially predict the object's position for the occluded frame [12, 9]. While this may allow the tracker to recover from transient occlusions, it is of little help during an extended occlusion where the object undergoes unpredictable trajectory changes.

By using multiple cameras, one can improve tracking performance. Previous work has handled occlusion by using multiple cameras spread over a large area to see around objects [2, 15] with the hope that at least one camera can detect a reliable match. However, if all cameras are significantly occluded, such as shown in Figure 1, tracking can still fail.

In this paper, we take a different approach. Instead of relying on a few of the cameras to see around an occluder our contribution is to use all cameras in a dense multi-camera setup to see *through* the occluder. This setup is similar to a "camera array" discussed in recent literature [14, 8, 23, 11], where a collection of cameras share a common working volume with all cameras working together as one "virtual" camera. The difference between our setup and a typical array is that we do not require regular spacing.

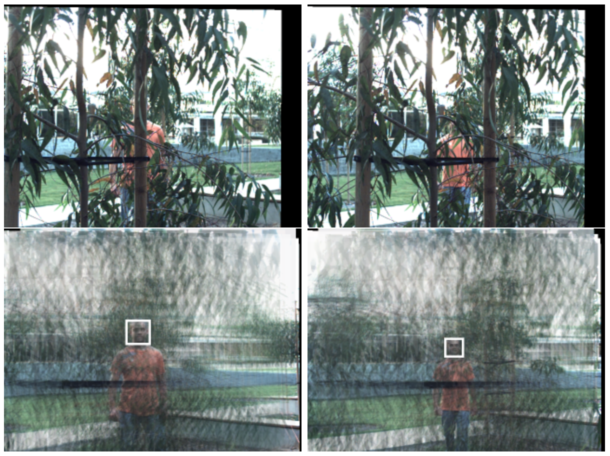
Figure 1. Tracking through occlusion. Frames from a single camera from a video sequence of a person moving behind a tree (top row). Using our method we can track the person successfully (bottom row). We track across the entire 360 frame sequence, which includes 200 straight frames where the person is heavily occluded.

The key insight is that when observing a scene using a dense multi-camera setup, different parts of the object to be tracked are occluded in each camera view. We leverage this property and aggregate data across cameras to see through occluders. In recent literature, this property has been leveraged by synthetic aperture photography [14, 8, 23]. In synthetic aperture photography, one aligns camera images to a plane and averages them together to approximate a camera with a very large aperture. These synthetically constructed images have very shallow depth of field and they blur out occluders significantly when focused on an object. While synthetic aperture has been used previously for seeing through occluders, to the best of our knowledge this is the first time tracking has been performed using this data.

The advantages our our approach are that we can track even when each camera observes very little of the target object – we will show that our system can successfully track an object when upwards of 70% of the object is occluded in every camera view. Our method does not perform any explicit modeling of the scene, allowing it to track in complex, dynamic scenes with moving cameras.

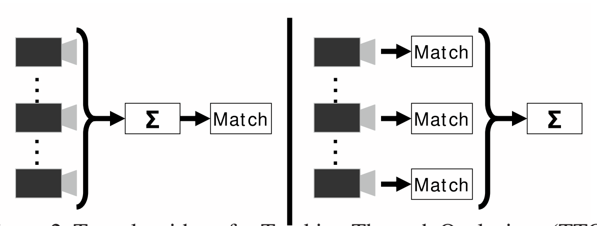

Figure 2. Two algorithms for Tracking Through Occlusions (TTO) using a dense multi-camera setup. The key difference between the methods is the relative order of aggregation and matching. The linear method first aggregates the data and then matches, while the non-linear method matches first to detect occlusions and then aggregates (right).

A further advantage of our setup is that by using all cameras together as a more powerful virtual camera, one can draw upon the extensive body of single-camera tracking algorithms, as opposed to using the smaller, more specialized collection of multi-camera methods. In this paper, we show logical extensions of eigenspace tracking [1]. We present two methods, where the key difference between the two is the relative ordering of matching and data aggregation across cameras.

We have explicitly chosen not to exploit temporal coherence or use a dynamics system for predicting object location as we do not want to obscure the contribution of using our setup for tracking through occlusion. We instead treat tracking as repeated detection so that it is clear that our methods enable matching on each frame of sequences that contain long periods of heavy occlusion.

We show results for a number of scenes filmed indoors and outdoors with an 8-camera setup and show a comparison of tracking accuracy versus occlusion density and number of cameras used. In the next section, we will discuss the previous work in this area, followed by an explanation of our tracking algorithms and implementation specifics. Lastly, we present results in Section 5 and a performance analysis in Section 6.

2. Related Work

While there is a large body of literature on the problem of object tracking from a single video camera, there is significantly less work on the use of multi-camera setups for tracking and, to the best our knowledge, none that uses dense multi-camera setups or camera-arrays.

The problem of occlusion, that is central to our work, is addressed in several different ways. In the case of single-camera tracking, one can treat the problem implicitly or explicitly. Implicit methods use filtering techniques such as Kalman filtering [12] or particle filtering [9] to predict the location of the object being occluded. Explicit methods often use a generative model that explains the video sequence in terms of multiple layers. Jojic *et al.* [10] use a variational expectation maximization algorithm to learn a mixture of sprites from the video sequence. Zhou *et al.* [27] maintain a list of foreground and background layers and computes a Maximum a Posteriori (MAP) estimate that best explains the observed image sequence in terms of shape, appearance, motion and layer ordering. Smith *et al.* [21] use a Bayesian framework to break an image into layers by tracking edges between frames. The edge information, in association with a Markov Random Field prior, gives the relative depth ordering of the different motion layers, thus enabling the system to reason about occlusions. Zhao *et al.* [26] use the ground plane assumption to obtain 3D information from a single video stream that can then be used to reason about occlusions in tracking multiple overlapping people. Okuma *et al.* [18] use a detector, in tandem with the tracker to recover from occlusions. Wu *at al.* [24] incorporate an extra hidden process for occlusion into a dynamic Bayesian network, to explain the observed image and to explicitly model occlusions. Xiao *et al.* [25] use a graph cut algorithm to model occlusions explicitly, a work that is inspired by the approach taken by Kolmogorov *et al.* [13] to reason about visibility in 3D reconstruction from multiple cameras.

Multi-camera setups solve the occlusion problem by integrating information across multiple cameras. For example, Mittal *et al.* [16] presented a system to segment, detect, and track multiple people using multiple cameras that uses intensity information to perform single-view pixel classification and derive 3D location. Occlusion is handled by the construction of a likelihood map on the ground plane that helps resolve visibility ambiguities. Fleuret *et al.* [3] use images from multiple cameras to create a virtual top view from binary motion detector blobs on which they compute an occupancy map. It is not clear how such multi-camera systems can be deployed in a (geometrically) complex outdoor scene (e.g., a forrest).

Another related line of research is the use of a stereo camera for object tracking [17, 5, 20]. In this case, depth is typically used as another channel and tracking is performed on a 4 channel image consisting of R, G, B, and depth. This is very different from our approach as we do not compute depth as an intermediate step, but rather look *through* the occluder whereas traditional stereo algorithms might find it difficult to obtain useful and reliable depth estimates in these regions. This difficulty was empirically studied by Vaish et al. [22], who investigated match metrics for finding stereo correspondence in the presence of occlusion. They showed that stereo reconstruction performance falls off as occlusion increases, with generally poor results with greater than 50% occlusion; most of the scenes we are concerned with have greater than 50% occlusion.

3. Tracking Through Occlusions

In this section, we present two algorithms for tracking trough occlusions using a calibrated dense multi-camera

Algorithm 1 Linear TTO

```
Initialization for frame $j = 0$
  • For the user specified 2D window on the first frame, compute the initial disparity automatically
  • For this disparity, compute the synthetic aperture image, this is the first basis image in the eigenspace
  • Initialize *occluded* flag to be unoccluded
For frames $j = 1$ to $k$
  • Compute synthetic aperture images for a range of disparities centered around the disparity of the previous frame
  • For each of these images, compute match scores to the eigenspace for windows centered in the 2D search area
  • Set the new position to be the location $(x, y, d)$ with the maximum match score
  • Scale the tracker window and basis images to account for any change in depth
  • Project the new window to each camera to get $N$ images of the target for the current frame, compute the mean (synthetic aperture image) and variance across these images
  • If the variance is low, the object is unoccluded, therefore update the eigenspace with the synthetic aperture image of the target
```

Figure 3. Linear TTO.

setup (we will specifically address calibration in Section 4). The key difference between our two methods: Linear Tracking Through Occlusions (TTO) and Non-Linear Tracking Through Occlusions, is the order of matching and aggregation in our pipeline, as shown in Figure 2. For both methods, the user must mark an initial 2D window on the central camera's image, to specify the location of the object to track. We then use a plane-sweep method similar to the method used by Joshi et al. [11], to find the initial depth value. Starting from this initial 3D location, we track the object using an eigentracking approach – we perform matching to a linear subspace that is trained to capture the appearance of the target object. This model is learned online [19], which is discussed in Section 3.4. We will now describe our two tracking algorithms in detail.

3.1. Linear Tracking Through Occlusions

Our first method operates by matching on synthetic aperture images [14, 8, 23]. When one of these images is focused at the depth of a target object, any potential occluders at a different depth will be severely defocused (blurred). If this depth difference is significant, given the baseline of the cameras, the occluder can be seen through.

We track the object by computing synthetic aperture images for a range of depths centered around the depth for the previous frame. We then perform matching between the synthetic aperture images for a number of windows centered around the 2D location of the object in the previous frame and pick the 3D location of the object that matches best.

Specifically, let $\{I^i\}_{i=1}^N$ denote the set of N camera images captured at a given time instance. Let $\mathbf{f}(I^i, P_i, d)$ be a warping function that aligns an image I^i to a particular depth, d, relative to a central reference camera using the projection matrix, P_i, for the ith camera. Then the synthetic aperture image for a depth d is:

$$I_d = \frac{1}{N}\sum_i \mathbf{f}(I^i, i, d). \tag{1}$$

For $\tilde{u}$, the template eigenspace, we define a matching function, $\mathbf{T}(I, \tilde{u}, x, y)$, that returns a scalar score for a fixed size image window centered at image position (x, y) according to how well it matches the eigenspace. See Section 3.3 for details of the matching function.

Now, given a search range of finite 2D image positions and depths, $\Omega = (x_{[min,max]}, y_{[min,max]}, d_{[min,max]})$, our goal is to find the maximum match value, (x_m, y_m, d_m), in the 3D spatio-depth volume:

$$\begin{aligned}(x_m, y_m, d_m) &= \operatorname*{argmax}_{(x,y,d)\in\Omega}\{\mathbf{T}(I_d, \tilde{u}, x, y)\} \qquad (2)\\ &= \operatorname*{argmax}_{(x,y,d)\in\Omega}\{\mathbf{T}(\frac{1}{N}\sum_i\{\mathbf{f}(I^i, i, d)\}, \tilde{u}, x, y)\}.\end{aligned}$$

3.2. Non-Linear Tracking Through Occlusions

If we process the camera streams individually, we can handle occlusions more directly. When matching on each image *before* aggregating the data, our second method detects occlusion on a per-pixel, per-camera basis by considering outliers from the eigenspace as occlusions. The advantage being that we can handle very dense occlusion even with a small number of cameras. In the limit, even if only 1 of N cameras sees a certain part of the object, the algorithm will detect that area as a match. In contrast, the synthetic aperture method is more likely to miss the match as the data from the $N-1$ cameras that are occluded will dominate and result in a low matching score.

Just as with the previous method, given finite range search range, Ω, we seek to find the maximum match value in the 3D spatio-depth volume:

$$(x_m, y_m, d_m) = \operatorname*{argmax}_{(x,y,d)\in\Omega}\{\frac{1}{N}\sum_i\{\mathbf{f}(\mathbf{T}(I^i, \tilde{u}, x, y), i, d)\}\}. \tag{3}$$

Comparing equations 2 and 3 one can see that the only difference is the order of the matching operator $\mathbf{T}$ and the

```
Algorithm 2 Non-Linear TTO

Initialization for frame $j = 0$
  • For the user specified 2D window on the first frame, compute the initial disparity automatically
  • For this disparity, project the window to each camera to get $N$ images of the target and construct the eigenspace
  • Initialize $occluded$ flag to be unoccluded
For frames $j = 1$ to $k$
  • For each camera image, construct "score images" by computing match scores to the eigenspace for windows centered in the 2D search area
  • Compute the mean of these score images aligned for a range of disparities centered around the disparity of the previous frame
  • Set the new position to be the location $(x, y, d)$ with the maximum match score
  • Scale the tracker window and basis images to account for any change in depth
  • Project the new window to each camera to get $N$ images of the target for the current frame, compute the variance across these images
  • If the variance is low, the object is unoccluded, so update the eigenspace using all $N$ images of the target
```

Figure 4. Non-Linear TTO.

summation. To summarize, Linear TTO aligns images to a depth, sums, and then matches to an eigenspace trained on synthetic aperture images, while Non-Linear TTO, aligns to a depth, matches to an eigenspace trained on camera images, and then sums the match costs. Each method results with a match cost defined for a range of possible positions (image position and disparity) and picks the position with the best (highest) score. Refer to Figure 3 and Figure 4 for pseudocode describing each algorithm.

3.3. Robust Matching

In tracking, we aim to find the location in a image that minimizes the distance, in a robust way, to the eigenspace appearance model. Our matching function, $\mathbf{T}(I, \tilde{u}, x, y)$, computes a match score according to the distance from the eigenspace. We compute this score by robustly projecting image data to the eigenspace.

Specifically, given an image patch $\mathbf{x}$ (in vector form) and eigenvectors $\tilde{u} = [u_1, \ldots, u_k]$, we use iterative reweighted least squares (IRLS) and minimize:

$$O_W(\mathbf{x}) = ||\mathbf{w}\mathbf{u}\lambda - \mathbf{w}\mathbf{x}||^2, \quad (4)$$

where λ is the vector of coefficients resulting from projecting the data onto the eigenspace and $\mathbf{w}$ is a diagonal weight matrix that is at first set to identity and is recomputed for each iteration to minimize the Minimax robust norm [7]. This process down-weights outliers and thus occlusions. The weight matrix is updated as follows:

$$\mathbf{w_{ii}} = \begin{cases} 1 & |r_i| < \Theta \\ \frac{\Theta}{|r_i|} & |r_i| \geqq \Theta \end{cases}, \quad (5)$$

where $r = \mathbf{u}\lambda - \mathbf{x}$ is the residual vector from the previous iteration of solving the weighted least-squares problem. Θ was set empirically to 0.05 for image data in the range $[0, 1]$ and 0.5 for zero-mean, unit-variance normalized image data (normalization is discussed in the next section).

When this process terminates, we take the sum of the resulting weights as the the match score. As inliers receive a weight of 1 and outliers are down-weighted, the sum of the weights is a robust measure of the number of inliers. IRLS can be quite time consuming, however, we found that when using a binary weighting, i.e., $\mathbf{w_{ii}} = 1$ when $|r_i| < \Theta$ and 0 otherwise, the process can be accelerated. For Non-Linear TTO, representing occlusion as binary is a reasonable approximation. Thus we use binary weights for the non-linear method as a speed optimization. Furthermore, we use a coarse-to-fine strategy to increase the speed of the matching algorithm.

3.4. Online Learning in Occluded Scenes

We build an eigenspace model for our object online by adding image data for the object from tracked positions using incremental PCA [19]; we use the 16 most significant eigenvectors as our basis. There is one shared eigenspace updated using frames from all camera views. To limit drift in the online learning process, we store the template from the first frame for the central reference camera and, after each update, we find the orthogonal component of the first frame's template with the updated basis and then include this orthogonal component as an additional basis vector. This ensures that the template for the first frame is always in the basis [6].

To ensure that occluded pixels are outliers and therefore down-weighted during IRLS, the object and occluder must differ in appearance and the eigenspace should only capture the object's appearance. Thus, when performing online leaning, we only update the eigenspace when there is no occlusion. We assume that the first frame is unoccluded so we have at least one frame for training.

We determine whether the object is occluded by looking at the per-pixel variance across camera images for a particular frame versus that of the first frame. Since the disparity and 2D location for the first frame are chosen to align the target across all N cameras, the variance taken per-pixel across cameras should be relatively low. In subsequent frames, the variance can be high when the pixel is not well aligned due to occlusion or due to tracking to an incorrect depth. We consider any pixels with variance less than γ (we use $\gamma = 500$, variance data is on the order of $[0, 255]^2$) to be well-aligned and "occlusion-free". The number of occlusion-free pixels is stored for the first frame. For any subsequent frame this number is recomputed and if it is below a percentage threshold τ (we use $\tau = 95\%$) of the number of unoccluded pixels in the first frame, we consider the object "occluded".

Specifically, the per-pixel variance image for depth d is:

$$V_d = \frac{1}{N-1} \sum_i (\mathbf{f}(I^i, i, d) - I_d)^2. \qquad (6)$$

where I_d is given by equation 1. We update the *occluded* flag after tracking a frame k as such:

$$occluded_{k+1} = \left(\sum_{\Delta_k} \{V_d^k < \gamma\} < \tau \sum_{\Delta_1} \{V_d^1 < \gamma\} \right), \qquad (7)$$

where Δ_k is the tracker window for the current frame and Δ_1 and V_d^1 are the first frame's tracker window and variance image, respectively. We have found that this relatively simple measure works quite well.

We train our eigenspace and perform matching on RGB data. For Linear TTO, we operate on zero-mean, unit-variance normalized image patches. The motivation behind this is to correct for contrast loss that can occur due to blurred out occluders. Consider the simple case of a blurred black occluder in a synthetic aperture image, any pixel occluded in even one camera view, will be averaged with black in computing the synthetic aperture image. This "occluder haze" causes a loss of contrast; working in a normalized space counteracts this effect. This is not an issue for Non-Linear TTO, as occlusion is more binary and there is no blurred occluder during matching.

4. Implementation Details

Our method requires a calibrated dense multi-camera setup where the cameras share a common working volume. There are no particular constraints on layout, i.e., camera positions can be arranged with 1D, 2D, or 3D layouts, spacing can be regular or irregular, and the setup can move as long as the cameras remained fixed relative to each other. For the examples in this paper, we use a horizontal linear arrangement of 8 640x480 pixels (Bayer pattern) Basler cameras mounted with 3 inch spacing, as shown in Figure 5.

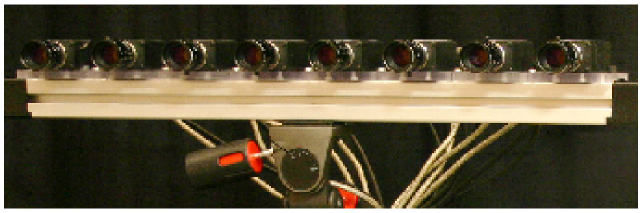
Figure 5. Our array of 8 VGA resolution video cameras.

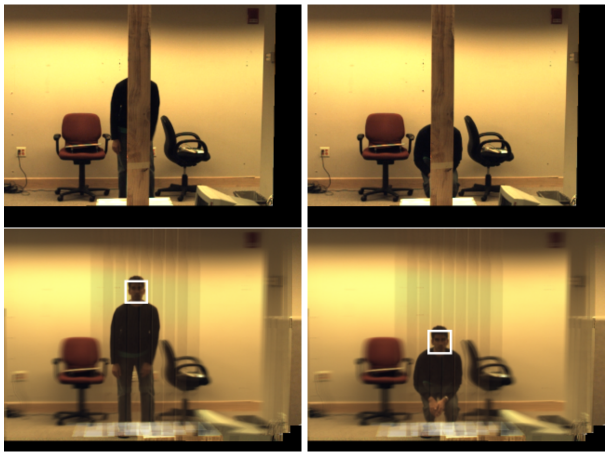
Figure 6. Person behind a wooden plank. Here we show two frames from a video sequence several seconds apart. From a single camera view (top row), the person is completely occluded. However, with synthetic aperture (bottom row), the plank blurs out and we can track him.

We geometrically calibrate our cameras (both extrinsics and intrinsics) using bundle adjustment after capturing corresponding points by tracking an LED. We choose a central camera as the reference camera and align relative to this camera. We perform basic color calibration by placing a Macbeth color checker in the scene so it is viewable by all cameras and then compute a color transform for each camera to match its image of the color checker to that of the central reference camera. We also perform vignetting calibration by imaging a constant intensity light-panel and computing per-pixel multipliers for each camera image to correct for falloff across the image plane.

The time complexity of our method is no worse than eigenspace methods on a per-camera basis. With our current system there is an additional time penalty as we process each video stream serially on a single CPU. Disk, memory I/O, and bus bandwidth limitations can add additional overhead. It would be a relatively straightforward extension to parallelize the processing of video streams using a multi-core CPU or a small cluster of PCs. Currently, tracking using RGB data from 8 cameras at 320×240 resolution with a 50×50 pixel target runs at 0.2 FPS using the linear method.

5. Results

In this section, we present results using our two algorithms. For each result we show two frames from significantly longer video clips. The reader is encouraged to view these clips and additional sequences online at http://vision.ucsd.edu/kriegman-grp/research/synthetic_ap_tracking/.

In Figure 6, we show frames from results using Linear TTO for a relative simple 200 frame sequence of a person walking behind a wooden plank. Tracking the person in this

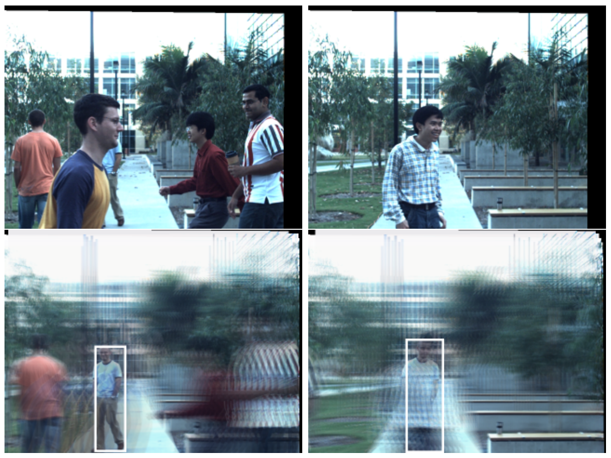
Figure 7. Tracking a person in a crowd. The person in a single camera is significantly occluded by different people in the group (top row). Using Non-Linear TTO we can track the person successfully (bottom row).

scene is impossible without the additional camera data, as from the single camera view, shown in the figure, the person is completely occluded. However, with synthetic aperture the plank blurs out and we can track the person.

In Figure 1, we show results using Non-Linear TTO on a 360 frame sequence of a person walking behind a tree outdoors. Of interest is that the view of the person in a single camera is significantly occluded by the tree, yet we are able to successfully track across the whole sequence. In addition, we panned the camera setup across the scene to keep the person in the field of view. This panning does not disrupt our algorithm even though it causes the background and occluders to change drastically from frame to frame. In our video available online, we show results for another tree sequence using both our methods. We further show that we can track this sequence using only 4 of 8 cameras using our second method, and we can track at both 320×240 and 160×120 resolution. In Figure 7, we show results using Non-Linear TTO for 100 frame sequence of a person walking in a plaza with several other people walking in front.

In Figure 8, we show results from Non-Linear TTO on a 90 frame sequence of a person moving as he waves a stuffed animal around in front of him. We track multiple objects by running our tracker independently for each object.

In Figure 9, we show results for a 200 frame sequence of dynamic random dot videograms with random dot occluders. This sequence was created from a planar square with a random dot pattern that was placed in front of a planar dynamic random dot background and behind a planar, 70% dense random dot occluder. Each object was warped appropriately according to the projection matrices for 8 virtual cameras in a linear arrangement according to the 3D location of the background, square, and occluder. We picked depths for the objects so that they would be similar to the real-world scenes we filmed. The occluder was effectively a

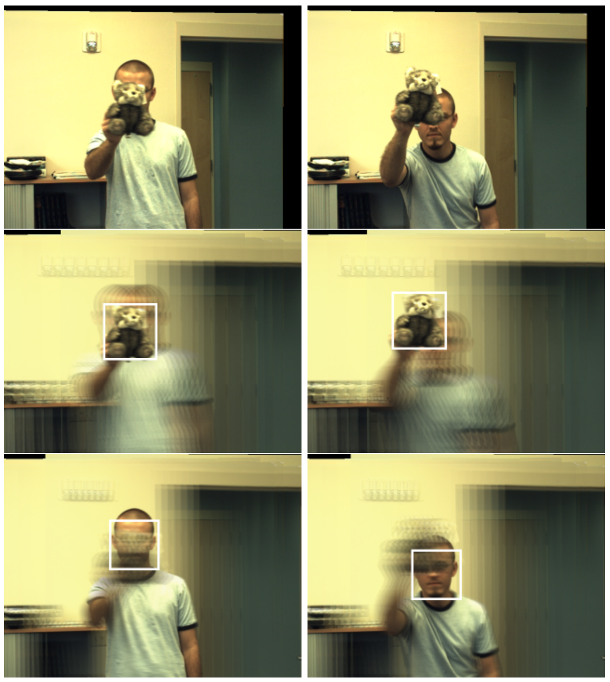
Figure 8. Tracking two objects. The person is occluded by the stuffed animal (top row). We track the person (middle row) and the occluder (bottom row) using Non-Linear TTO. Note that both the 2D position of the objects and relative depth have changed – in the second frame the person has moved the stuffed animal closer to his face.

couple meters from the cameras, the background was many meters away, and the square moved between these two. The occluder and background have a dynamic appearance and static locations. The square has a static appearance but moves in a spiral-like path away from the cameras. The results for this sequence were generated using Linear TTO. One interesting aspect of this sequence, in contrast with the previous sequences, is that tracking the object in one view would be quite difficult even for a human observer; however, our tracker successfully tracks the whole sequence.

6. Performance Evaluation

To analyze the performance of our two algorithms as a function of occlusion density and number of cameras, we constructed several datasets with synthetic occluders of varying density. We first tracked an unoccluded multi-camera sequence of a person walking around an office room to get "groundtruth" results and then used view interpolation to synthetically produce 20 virtual camera streams. We combined these data streams with randomly generated synthetic planar occluders translated appropriately according to their fixed depth – about a meter from the virtual cameras.

Using this process, we created datasets similar to that in Figure 1 except with a regular synthetic leaf-like occluder at three different densities: 50%, 70%, and 90%. We then ran our methods using subsets of sizes $[2, 4, 6, 8, 12, 16, 20]$ of

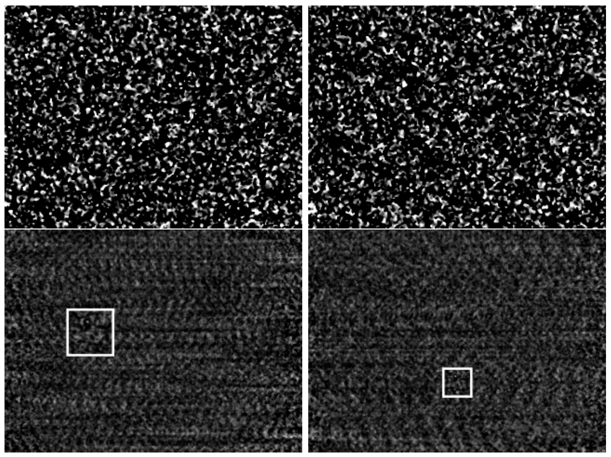
Figure 9. Random dot videograms with random dot occluders. The background and 70% dense random dot occluder have dynamic texture. A small (statically) textured square moves between them. The view from a single virtual camera several seconds apart (top row). The tracked locations using Linear TTO (bottom row). The tracked locations are within 1% of groundtruth.

the 20 cameras. The cameras were selected by always using the right and left-most cameras, to ensure the baseline was fixed across trials, and equally spacing the cameras selected between the two extreme positions. Figure 10 shows results for these data sets. In these plots, we show the percentage of a 180 frame sequence that was correctly tracked. Our results show that the ability to track with synthetic aperture is highly dependent on the number of cameras and at higher-levels of occlusion the method performs quite poorly with a small number of cameras. Note, however, that at 70% occlusion we still track 100% of the frames with only 8 cameras. The results for Non-Linear TTO shows that this method is much more robust with smaller numbers of cameras, and it tracks 100% of the frames with only 2 cameras for 50% and 70% occlusion.

7. Discussion and Future Work

We showed how to track objects in the presence of significant occlusion using Linear and Non-Linear TTO. The linear method is very efficient, as the time consuming operation of matching is performed on a single video stream. The main advantage of the non-linear method is that it can handle very dense occlusion, even with a small number of cameras, at the cost of increasing the number of matching operations linearly in the number of cameras. Having analyzed the performance of the two methods, we conclude that when there is significant occlusion and a large number of cameras at one's disposal, Linear TTO is preferred due to its speed. When fewer cameras are available, Non-Linear TTO is preferred.

An interesting benefit of our work is its scalability – one could imagine building improved multi-camera systems, such as room-sized tracking systems [2, 15] where each camera is itself a synthetic aperture tracking camera.

Our methods are robust and handle a wide variety of situations; however, some limitations remain. We cannot track the object if the cameras' baseline is not large enough to "see around" an occluder given its depth, width, and the distance from it to the object. Also, when the object is too close to the occluder or the occluder is too wide relative to the baseline our methods could fail.

As shown in Section 6, the density of the occluder affects the number of cameras needed, and, as a general rule, more cameras are needed as occlusion density increases. We showed that our linear method can track in 70% occlusion with 8 or more cameras, while our non-linear method can track with 70% occlusion with even 2 cameras. These methods cover a large range and can track as long as there is enough appearance difference between occluder, object, and background. In the case of ambiguous appearance, e.g., a scene with similarly colored and textured objects, our methods can fail.

Another limitation of our current algorithm is that we do not update the eigenspace in the presence of occlusion. As a result, if significant appearance changes occur when the object is occluded, the tracker will drift. We do not update the model in this case to avoid errantly adding the occluder to the model; however, for the non-linear method, as we do have a mask for the occluder, we could, in principle, update the model using only the unoccluded pixels. However, since the mask is likely to have some errors, performing this update is not trivial. This is something we are investigating.

Another extension to our work is to improve our tracking results by using more sophisticated tracking methods including dynamics models and sampling techniques. We believe that a strength of this work is that simple methods can be very powerful even without models; nevertheless, we have no doubt that our results could be improved by adapting more sophisticated tracking algorithms.

Lastly, we think there are many interesting situations left to be explored using multi-camera systems, such as tracking in inclement weather or tracking objects behind glass windows with strong reflections; we are very interested in exploring these scenarios.

8. Acknowledgements

We would like to thank the anonymous reviewers for their comments. We are very grateful to the willing students who appeared in our videos. This work was partially completed while the first author was an intern at MERL; he was additionally funded by NSF grant DGE-0333451.

References

[1] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *ECCV (1)*, pages 329–342, 1996.

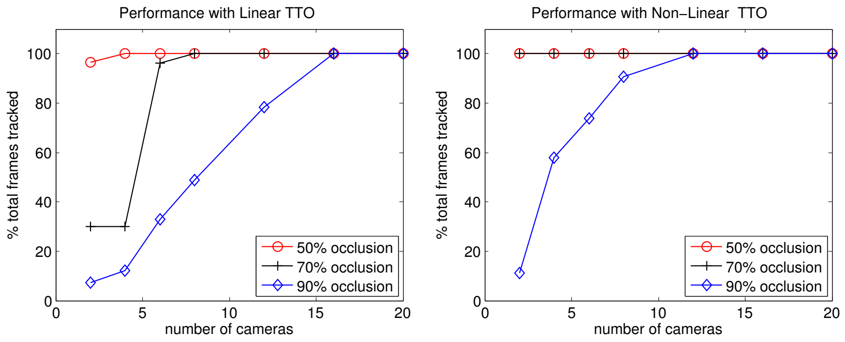

Figure 10. Tracking robustness as a function of occlusion and number of cameras. Linear TTO (left). Non-Linear TTO (right).

[2] Q. Cai and J. Aggarwal. Tracking human motion using multiple cameras. In *Proc. of Intl. Conf. on Pattern Recognition*, 1996.

[3] F. Fleuret, R. Lengagne, and P. Fua. Fixed point probability field for complex occlusion handling. In *IEEE International Conference on Computer Vision (ICCV)*, pages 694–700, 2005.

[4] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(10):1025–1039, 1998.

[5] M. Harville, A. Rahimi, T. Darrell, G. G. Gordon, and J. Woodfill. 3d pose tracking with linear depth and brightness constraints. In *ICCV*, pages 206–213, 1999.

[6] J. Ho, K.-C. Lee, M.-H. Yang, and D. Kriegman. Visual tracking using learned linear subspaces. *cvpr*, 01:782–789, 2004.

[7] P. J. Huber. *Robust Statistics*. John Wiley and Sons, New York, 1981.

[8] A. Isaksen, L. McMillan, and S. J. Gortler. Dynamically reparameterized light fields. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 297–306. ACM Press/Addison-Wesley Publishing Co., 2000.

[9] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[10] N. Jojic and B. J. Frey. Learning flexible sprites in video layers. In *Proc. of Intl. Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 199–206, 2001.

[11] N. Joshi, W. Matusik, and S. Avidan. Natural Video Matting using Camera Arrays. *ACM Transactions on Graphics*, 25(3), July 2006.

[12] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, 82:35–45, 1960.

[13] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part III*, pages 82–96, London, UK, 2002. Springer-Verlag.

[14] M. Levoy and P. Hanrahan. Light field rendering. In *Computer Graphics*, SIGGRAPH 96 Proceedings, pages 31–42, New Orleans, LS, Aug. 1996.

[15] A. Mittal and L. Davis. Unified multi-camera detection and tracking using region matching. 2001.

[16] A. Mittal and L. Davis. M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene. *International Journal of Computer Vision*, 51(3):189–203, 2003.

[17] L.-P. Morency, A. Rahimi, N. Checka, and T. Darrell. Fast stereo-based head tracking for interactive environments. *Face and Gesture Recognition*, 00:0390, 2002.

[18] K. Okuma, A. Taleghani, N. de Freitas, J. Little, and D. Lowe. A boosted particle filter: Multitarget detection and tracking. In *European Conference on Computer Vision*, pages 28–39, 2004.

[19] D. A. Ross, J. Lim, and M.-H. Yang. Adaptive probabilistic visual tracking with incremental subspace update. In T. Pajdla and J. Matas, editors, *Proc. Eighth European Conference on Computer Vision (ECCV 2004)*, volume 2, pages 470–482. Springer, 2004.

[20] D. B. Russakoff and M. Herman. Head tracking using stereo. *Mach. Vision Appl.*, 13(3):164–173, 2002.

[21] P. Smith, T. Drummond, and R. Cipolla. Layered motion segmentation and depth ordering by tracking edges. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(4):479–494, 2004.

[22] V. Vaish, M. Levoy, R. Szeliski, C. L. Zitnick, and S. B. Kang. Reconstructing occluded surfaces using synthetic apertures: Stereo, focus and robust measures. In *CVPR 06*, pages 2331–2338, 2006.

[23] V. Vaish, B. Wilburn, N. Joshi, and M. Levoy. Using plane + parallax for calibrating dense camera arrays. In *Proceedings of CVPR04*, 2004.

[24] Y. Wu, T. Yu, and G. Hua. Tracking appearances with occlusions. In *Proc. of Intl. Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 789–795, 2003.

[25] J. Xiao and F.-M. Shah. Motion layer extraction in the presence of occlusion using graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1644–1659, 2005.

[26] T. Zhao and R. Nevatia. Tracking multiple humans in complex situations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1208–1221, 2004.

[27] Y. Zhou and H. Tao. A background layer model for object tracking through occlusion. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 1079, Washington, DC, USA, 2003. IEEE Computer Society.