3D Occlusion Recovery using Few Cameras*

Mark Keck James W. Davis
Dept. of Computer Science and Engineering
The Ohio State University
Columbus, OH 43210
{keck,jwdavis}@cse.ohio-state.edu

Abstract

We present a practical framework for detecting and modeling 3D static occlusions for wide-baseline, multi-camera scenarios where the number of cameras is small. The framework consists of an iterative learning procedure where at each frame the occlusion model is used to solve the voxel occupancy problem, and this solution is then used to update the occlusion model. Along with this iterative procedure, there are two contributions of the proposed work: (1) a novel energy function (which can be minimized via graph cuts) specifically designed for use in this procedure, and (2) an application that incorporates our probabilistic occlusion model into a 3D tracking system. Both qualitative and quantitative results of the proposed algorithm and its incorporation with a 3D tracker are presented for support.

1. Introduction

Detection and tracking are two very important problems to the surveillance community in the computer vision field. It is widely known that one of the biggest challenges when dealing with these problems is handling occlusion. The presence of occlusion in imagery will often lead to poorer detection performance [16]. Similarly in tracking, if the tracked object moves behind an obstruction, the tracker will often lose the object completely and not be able to recover.

Because occlusions can be such a hindrance to trackers, there have been many methods proposed in the literature for dealing with them. Some trackers focus on reasoning about *dynamic* occlusions [7, 10, 15], which occur when one foreground object occludes another foreground object. However, these trackers tend to have difficulty with full *static* occlusions (*i.e*. when a foreground object is occluded by a background object) because they are often based on background subtraction.

*Appears in *IEEE Conf. on Computer Vision and Pattern Recognition*, Anchorage, AK, June 2008

Another way to deal with occlusion is to add multiple views to a scene and track in 3D. 3D tracking affords many advantages. For example, 3D tracking offers a way to register a real-world scene with a virtual model (*e.g*. georegistration). However, even if cameras are deployed wisely, trackers that reason well about dynamic occlusions still suffer when static occlusions occur in multiple views.

This inadequacy necessitates the modeling of static occlusions for tracking systems. In this work we propose a method for recovering a static 3D occlusion model for a particular scene. We assume a wide-baseline configuration of few ($= 3$) static cameras because surveillance systems can rarely afford to put a large number of cameras in a single area, especially with small baselines. We refer to this as our real-world assumption. To deal with this small number of cameras, we do not attempt to reconstruct the *occluding structures*, but instead detect *occluded areas* in 3D, which, as we will show, is still robust even with few cameras.

Specifically, our proposed algorithm performs an iterative procedure which learns the occluded areas in each of the camera views by first solving the voxel occupancy problem via graph cuts. We introduce a novel energy function in the graph cut formulation that is specially designed to reconstruct foreground objects that are occluded in some camera views by utilizing the learned occlusion model. This solution to the voxel occupancy problem is then used to update the occlusion model for the next iteration, similar to the EM algorithm. We finally show an application that incorporates the occlusion model into a tracking algorithm to improve results.

The rest of the paper is organized as follows. An in-depth investigation of related work is supplied in Sect. 2. An overview of the algorithm is provided in Sect. 3. We describe our experimental evaluation and discuss results in Sect. 4. We offer concluding remarks and future directions for this research in Sect. 5.

2. Related Work

Occlusion recovery is a problem that has been studied in both cognitive and computer vision fields. Classically occlusions in images have been identified by finding T-junctions, popularized by [2]. T-junctions in single images are a very powerful cue for finding occluding contours and are still often employed to recover occluding boundaries in multiple cameras [1, 3]. As strong as T-junctions can be, practical implementations are still a challenge as reliable extraction of the features and matching among multiple views (especially wide-baseline views) are both still difficult problems.

Because of this, other cues have also been used to find occlusion boundaries and regions in images, such as in [8] where the *effective boundary* of an object is defined and evidence is accumulated over time to find the exact occluding boundary. These occluding regions are found in image space instead of 3D, and are used to reconstruct visual hulls [12] in the presence of partial occlusions. Further, in [9] these same authors fully recovered the 3D structure of occlusions in the scene by modeling $p(\mathcal{O}|\mathcal{I},\mathcal{B})$, the posterior probability of a voxel being an occluder given a video sequence $\mathcal{I}$ and a background model $\mathcal{B}$. Occlusions are discovered by maximizing this posterior. Although [9] has a similar goal to the proposed work, it isn't exactly the same. The authors there are reconstructing *occluding structures*, which is possible with a large number of cameras (9 in their experiments), while we are only modeling the probability that a particular voxel is *being* occluded in a particular view, which is feasible with few cameras.

We think it is also important to cite relevant tracking algorithms that explicitly model occlusions. Some algorithms like [7, 10, 15] are more concerned with dynamic occlusion reasoning than recovering static occlusions. Other algorithms model static occlusions for object tracking (*e.g.* [21] uses motion layer estimation to find occluding layers), but generally these approaches are done in image space and are difficult to extend to 3D.

To overcome some of these limitations we propose an algorithm which will directly model occlusions in a 3D space. The algorithm is an iterative procedure that at each frame first solves the voxel occupancy which then feeds back into the system by updating the occlusion model. Because the algorithm is not attempting to reconstruct the occluding objects, and only modeling which areas are occluded and which are not, over time it is able to reliably identify these areas, even with very few cameras.

3. Algorithm

We start by defining our problem mathematically. We assume a scene with M overlapping views that are calibrated. We then generate a voxel lattice in the calibration space that has dimensions $N = n_x \times n_y \times n_z$ and enumerate the voxels from 1 to N.

With this voxel lattice, we define a binary random variable $\mathcal{O}_j^i$ over the lattice that takes on value 1 if voxel j is occluded when projected into view i and 0 otherwise. We denote the probability over the random variable as $P(\mathcal{O}_j^i = b)$, where $b \in \{0, 1\}$ is a binary constant. In the remainder of this paper, for brevity we will use the notation $P(\mathcal{O}_j^i)$ for $P(\mathcal{O}_j^i = 1)$ as it is obvious that $P(\mathcal{O}_j^i = 0) = 1 - P(\mathcal{O}_j^i)$. Our goal is to estimate this distribution per voxel to determine which voxels are occluded in each view.

We estimate these probabilities using an iterative, EM-style framework. The algorithm operates on a video sequence, and at each incoming frame, it first solves the voxel occupancy problem via a specialized graph cut approach which takes into account the current probabilistic occlusion model. We then update the occlusion model using this solution. We specify details in the following subsections.

3.1. MRF Solution with Occlusion Model

In this section we discuss our solution to the voxel occupancy problem, which is an MRF formulation that takes advantage of the explicit occlusion model.

We first remind the reader of the voxel occupancy problem. The goal is to determine which voxels in a lattice are occupied at a given frame. It is analogous to the foreground-background segmentation problem in 2D space. The problem is often solved by finding the visual hull [12], and over the past five years graph-based solutions like [19] have become more popular as they allow fuzzy local decisions that are resolved by a global energy minimization. The minimization is done by constructing a Markov Random Field (MRF) and finding the min-cut on the graph which separates the voxels into two mutually exclusive sets, one which corresponds to the "on"-voxels and one which corresponds to the "off"-voxels. This approach has two advantages: (1) it does not force the algorithm to make difficult decisions at each voxel, and (2) it naturally incorporates dependence of neighboring voxels into the energy minimization formulation.

When solving a labeling problem with an MRF, the typical approach is to maximize the posterior distribution of the labeling given the data, denoted $P(\mathcal{L}|\mathcal{D})$, by recognizing that it is proportional to the likelihood of the data times the prior of the labeling:

$$P(\mathcal{L}|\mathcal{D}) \propto P(\mathcal{D}|\mathcal{L})P(\mathcal{L}) \tag{1}$$

The optimal labeling $\mathcal{L}^*$ is defined as the labeling which maximizes the product

$$\mathcal{L}^* = \arg\max_{\mathcal{L}} P(\mathcal{D}|\mathcal{L})P(\mathcal{L}) \tag{2}$$

The heart of the problem then lies in modelng these two distributions, the first being the likelihood of the data given

the labeling, and the second being the prior of the labeling $P(\mathcal{L})$. This is often transformed into negative log-likelihood space for numerical stability:

$$\mathcal{L}^* = \arg\min_{\mathcal{L}} -\ln P(\mathcal{D}|\mathcal{L}) - \ln P(\mathcal{L}) \quad (3)$$

We now describe how we model these terms in our framework. We would like to note that there are some similarities between our approach and the approach to foreground-background segmentation in [17], as we both model foreground and background likelihoods for input into an MRF. However, the proposed energy function is different to account for our explicit occlusion model, and further our segmentation problem is in 3D, not image space.

3.1.1 Voxel Likelihood

To estimate the likelihood of the observed data given a particular voxel, we model both the foreground and background likelihoods in the images where that voxel projects. We create a background model $\mathcal{B}^i$ where $i = 1, \ldots, M$ for each camera view. In this work we assume that the background is a single Gaussian in RGB color space with a diagonal covariance matrix. Therefore the likelihood of a voxel being generated by this distribution given that its label is 0 (*i.e.* $\mathcal{L}_j = 0$) is given by:

$$P_B(\mathbf{d}_j^i|\mathcal{B}^i) \propto \exp\left(-\frac{1}{2}(\mathbf{d}_j^i - \mathbf{b}_j^i)^\top \Sigma^{-1}(\mathbf{d}_j^i - \mathbf{b}_j^i)\right) \quad (4)$$

where $\mathbf{d}_j^i$ is an RGB 3-vector in image i where voxel j projects, and $\mathbf{b}_j^i$ is the RGB 3-vector to the corresponding pixel in the background model for image i. We can also model the likelihood that a voxel belongs to the foreground. We will model this as a uniform distribution, intuitively meaning that we assume all colors are equally likely to appear as part of a foreground object. We denote this uniform distribution as constant $\gamma = 1/(R \times G \times B)$ where R, G, and B are the number of possible colors in each of the bands (in this case $R = G = B = 256$).

Up to this point we have ignored the information we have from the occlusion model. We can improve our estimate of the foreground likelihood with this information. If it is known that a voxel is very likely occluded in image i, then it is also highly probable that the projection of that voxel into image i will resemble the appearance of the background model, *even if the voxel is truly occupied*. We can incorporate this knowledge into our foreground model in the following way:

$$P_F(\mathbf{d}_j^i|\mathcal{F}^i, \mathcal{B}^i) = P(\mathcal{O}_j^i)P_B(\mathbf{d}_j^i|\mathcal{B}^i) + \left[1 - P(\mathcal{O}_j^i)\right]\gamma \quad (5)$$

where $P_B(\mathbf{d}_j^i|\mathcal{B}^i)$ is given by Eqn. 4. By the same reasoning, the occlusion model will not alter the background likelihood function P_B. We incorporate this occlusion model expecting that in cases where we think something is very likely occluded in a given view, it can still be labeled as foreground if the other views label the voxel as likely to be in the foreground.

With these two likelihood functions, we can rewrite the first factor of Eqn. 2 as:

$$P(\mathcal{D}|\mathcal{L}) = \prod_j^N \prod_i^M P_B(\mathbf{d}_j^i|\mathcal{B}^i)^{1-\mathcal{L}_j} P_F(\mathbf{d}_j^i|\mathcal{F}^i, \mathcal{B}^i)^{\mathcal{L}_j} \quad (6)$$

3.1.2 Labeling Prior

To model the prior in Eqn. 2, in these types of image labeling problems one often chooses a predefined prior $P(\mathcal{L})$. A popular choice is the Ising model, which preserves discontinuities while still filling in the noisy gaps of the reconstruction:

$$P(\mathcal{L}) = \exp\left\{\sum_{i<j} \lambda\left[\mathcal{L}_i\mathcal{L}_j + (1-\mathcal{L}_i)(1-\mathcal{L}_j)\right]\right\} \quad (7)$$

where i and j are neighboring nodes in the graph and λ is a system parameter, usually a small positive number. In our case on the lattice, these are the six-connected neighbors in the x, y, and z directions. This model gives a higher likelihood to labelings that are smooth (*i.e.* neighbors usually have the same label). However, it is limited as it gives all connections the same weight, namely λ. We again could improve this function by including scene knowledge. Specifically, we would like to increase the likelihood of assigning two voxels the same label when they are neighbors and they are in the "same" image region. We adopt this idea because we don't want these occluded regions to bleed through the boundaries of edges of the objects in the images, as it is likely that an occluding region will have the same color/texture since the region is likely a single object. To this end, we only allow this increase in likelihood if two regions are part of the same object in a particular image.

To define "same", we perform an image segmentation on the background image. We first perform a nonlinear smoothing of the image based on [13], extended for color images. This will result in a smoothed color image that respects edges and does not smooth over them. We then perform a mean shift segmentation[5] on the smoothed image.

Our modified Ising model then becomes

$$P(\mathcal{L}) = \exp\left\{\sum_{i<j}\left[\sum_k^M \lambda\delta(\mathcal{S}_i^k, \mathcal{S}_j^k)\right]\right\} \quad (8)$$

There are several symbols to define for this equation. First, λ is again a small positive constant. The function $\delta(\cdot,\cdot)$ is the delta function, which returns 1 if the two arguments are

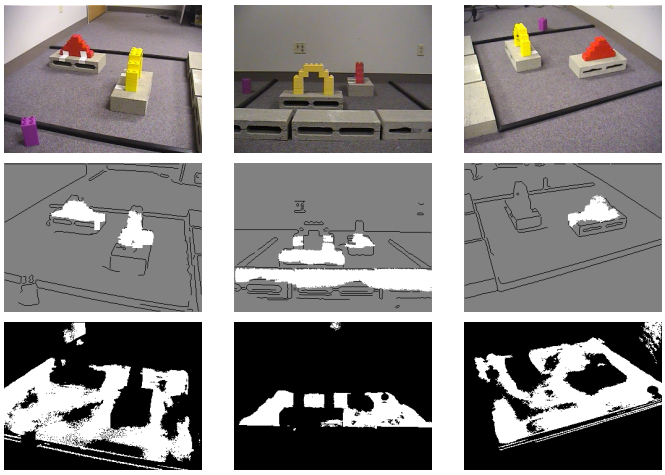

Figure 1. Top row: the layout of the BALL experiment. There are two large obstructions in the scene we wish to extract. Middle row: results of our method. Bottom row: cumulative background subtraction results on this sequence.

the same and zero otherwise. Finally, $\mathcal{S}_i^k$ is the segment in image k to which the projection of voxel i belongs.

Now we can rewrite Eqn. 3 in terms of our likelihood and prior.

$$\mathcal{L}^* = \arg\min_{\mathcal{L}} \left[\sum_i^N E^i(\mathcal{L}_i) + \sum_{i<j} E^{ij}(\mathcal{L}_i, \mathcal{L}_j) \right] \quad (9)$$

where

$$E^i(\mathcal{L}_i) = \sum_j^M -\ln P(\mathbf{d}_j^i|\mathcal{B}^i)^{1-\mathcal{L}_j} P(\mathbf{d}_j^i|\mathcal{F}^i, \mathcal{B}^i)^{\mathcal{L}_j} \quad (10)$$

and

$$E^{ij}(\mathcal{L}_i, \mathcal{L}_j) = -\left[\sum_k^M \lambda\delta(\mathcal{S}_i^k, \mathcal{S}_j^k) \right] \quad (11)$$

We write these in the form of Eqn. 9 to show that this log-likelihood indeed an element of the $\mathcal{F}^2$ class of energy functions defined in [11]. It also satisfies the regularity condition. Given this, we can optimally solve our labeling problem via graph cuts. We use the graph construction provided in [11] to build the graph based on our energy function, and minimize it using the standard push-relabel max flow algorithm to find the min-cut solution to our voxel occupancy problem.

After solving the voxel occupancy problem, we perform a 3D connected component algorithm and remove small components (those with less than 20 voxels) to remove spurious noise, which is analogous to removing small regions after background subtraction. We would also like to point out that we are reconstructing the visual hull in the presence of partial, and even full, occlusion in the imagery, solving the same problem as [8] in a different manner.

3.2. Occlusion Model Update

With the solution to the voxel occupancy problem we can update our occlusion model. Up to this point, we have just assumed that we have some occlusion model $\mathcal{O}$ at time t. Specifically what we store are two numbers. The first is the value $P(\mathcal{O}_j^i)$ for each voxel j in each view i, given a total of $N \times M$ probabilities.

We also keep track of the number of times a particular voxel has been labeled as "on" by the min-cut algorithm, which we will denote α_j for each voxel j. Given these two numbers we update $P(\mathcal{O}_j^i)$ as follows. We first get a foreground mask by thresholding the background subtraction results for each frame. We then project each "on" voxel from our min-cut solution into these foreground masks. For each projection, if the voxel j projects to an "on" pixel in foreground i, then we update by the rule

$$P(\mathcal{O}_j^i) = \frac{P(\mathcal{O}_j^i)\alpha_j}{\alpha_j + 1} \quad (12)$$

and if it projects to an "off" pixel, we assume that voxel j is occluded in view i, and we update the occlusion model by the rule

$$P(\mathcal{O}_j^i) = \frac{P(\mathcal{O}_j^i)\alpha_j + 1}{\alpha_j + 1} \quad (13)$$

After each of the occlusion probabilities are updated, we then increment α_j for each voxel labeled as "on". Intuitively, we are just keeping track of both the number of times the voxel has been occupied, and also how many of those times it was not in the foreground in each view. This estimates the probability with which we believe a voxel is occluded in each view.

This process then iterates over all frames of a sequence, incrementally improving the estimate of the occlusion with

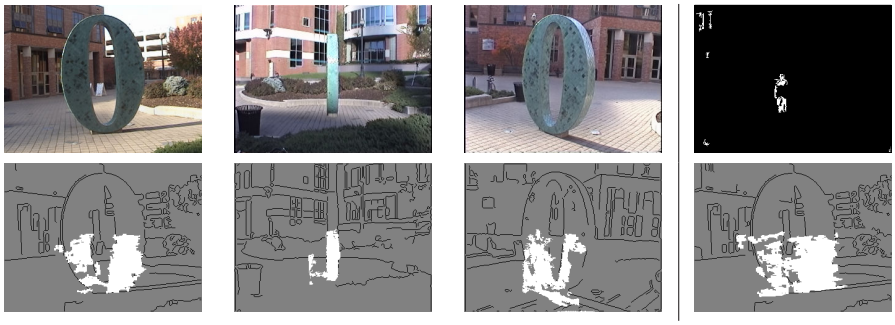

Figure 2. Top row: the layout of the BIG ZERO experiment. Bottom row: results on this video sequence.

every frame. We note it has a similar motivation to [4], as we refine our estimate across time due to the lack of cameras, except again our algorithm is not doing a full 3D reconstruction of the occluding structure. In the following sections we provide experimental results to support our method.

4. Experiments

To test the proposed method, we applied to three datasets, each of which consisted of a sequence of video captured from three Sony Handycam hand-held video cameras always at half (320×240) resolution. First we captured an indoor video sequence to show proof of concept. We then deployed the three cameras at two different locations outside on a university campus. Each of the two locations had at least one large occlusion visible in each view.

In all cases, we started by manually selecting around 30 points in each of the camera views to calibrate the scene using the method from [14]. We then created a voxel occupancy lattice in this calibrated space of dimension $n_x = n_y = n_z = 60$ giving a total of 216,000 voxels in each lattice. We implemented our min-cut solution by utilizing the max flow algorithm that is available in the Boost Graph Library [18].

4.1. Indoor BALL Experiment

The layout of our first experiment can be seen in Fig. 1. We have three views of a scene in which two large lego-built structures occlude different portions of each view. With the cameras deployed, we captured video through a Matrox Morphis Quad at a rate of $\sim$20 Hz for about six minutes, resulting in a total of 7630 frames. Because our model of occlusion is based on activity, we inserted a "Weazel Ball" (an autonomous pet toy) into the area and allowed it to roll around the scene on its own during the capture time. The learned occlusion model is shown in the middle row of Fig. 1. We created these images by raytracing each pixel and determining if that pixel intersected a voxel that has a high probability of occlusion (*i.e.* $P(\mathcal{O}_j^i) \geq .999$). We again did not allow 3D regions with size less than 20 voxels. If the pixel does intersect with such a voxel, it is colored white. Edges in the background image are colored black, and all other areas are colored gray.

One can see that most of the occluding structure is colored white and nothing else in the scene is found as an occlusion, but that some areas of the occluding structures are *not* recovered. However, one must keep in mind that our approach is driven by activity in the scene. When one considers that the ball is a rather small item (it is not as tall as the structures), we have recovered almost everything we can from the scene. Still, for a quantitative analysis of our results, we first reconstructed the visual hulls of the occluding structures to serve as ground truth, and then created per-view depth maps by raytracing this ground truth with the camera calibration matrices. Using this ground truth, we project all voxels that have been labeled as "on" at least ψ_{min} times into each of the three views. This will discount areas that have rarely been occupied by the ball. In our experiments $\psi_{min} = 20$.

With the voxels projected to corresponding pixels, we then compute the confusion matrix. We define a true positive as a voxel v that is labeled as occluded by our algorithm and is projected onto a pixel p_v^i in view i such that the depth map at p_v^i is less than the depth voxel v. True negatives, false positives and false negatives follow similarly. Again, we consider a voxel as labeled occluded in view i by our algorithm if $P(\mathcal{O}_v^i) \geq .999$. Doing this we get an F-measure of .8170, suggesting that we are effectively discriminating between occluded and visible areas when enough activity has taken place in the monitored scene.

For comparison to our results, we display binary images showing cumulative background subtraction results in the bottom row. Pixels that are white in this image are those pixels that were found in the foreground at least ψ_{min} times. The black pixels are therefore those pixels that either occluded something or were not active at least ψ_{min} times (the ball rarely rolled into that area). We can see that the areas found in our result images correspond to those areas that are black in the bottom row that correspond areas that

are both occluded and have had at least a minimal amount of activity, which supports our method.

4.2. BIG ZERO Sequence

To further demonstrate the effectiveness of the method, we also captured data from two outdoor scenarios. In the first scenario we set up three cameras around a large zero shaped structure on campus. The three cameras all have very different views of the structure, with one looking directly at it. The views can be seen in Fig. 2 on the top row. In this sequence we captured from the three cameras at real time, synchronizing the video post-capture, and ended up with 10620 frames.

The results are shown on the bottom row, with the white pixels being those which have some occlusion found along their line of sight. We can see in large part that even here in the outdoor scenario our algorithm is able to recover a great deal of the occluded areas that it possibly can, as the top part of the structure is not recoverable (there are no voxels behind it that are ever occupied in the entire sequence). However, as shown, there is more noise outdoors than indoors. This noise is attributed to two factors.

The first factor is the presence of shadows and highly varying lighting conditions. For instance in the lower right area of the third view, next to the large structure, there is some noise that shows up as false positives. This problem exists because, due to the location of the sun, the shadows of the people in the scene near the structure are consistently cast in that area of the image. These areas actually would have been removed, but they are too small for people to occupy. A similar effect happened in image 1, but the regions were small enough so that the connected component algorithm could remove them on the lower right side, and left only a small amount of noise on the lower left.

The other factor that significantly contributed to noise was poor background subtraction results in view two. A sample background image is provided on the top row on the far right of Fig. 2. There were many frames with this same effect, which led to the small amount of noise in image two (false positives), although some noise voxels were removed via connected components. This also had somewhat of an effect on the false negatives at the edge of image 3, where some of the zero structure was missed. This was because background subtraction failed in view 2 very often, and this particular area was usually occluded in both views 2 and 3. Therefore this area was very difficult to reconstruct.

On the other hand, the method is able to extract the large parts of the structure but leave out the inner area both in views 1 and 3, lending the method credibility even in an outdoor scenario with as few as 3 cameras. Note that the method is easily extendible to more than 3 cameras, but because we are primarily concerned realistic scenarios we have limited ourselves to 3. In the case of more cameras, we could only expect the results to improve as the visual hulls will become much crisper and produce less noise. We also illustrate the effectiveness of using the new smoothness term by comparing to Fig. 2, at the lower right. One can see that much of inside of the structure is marked as occluded, again because this model does not respect segmentation boundaries.

4.3. ARCHITECTURE Sequence

The final test location was chosen to see how the algorithm would perform in an outdoor area with multiple occlusions in each view during a time of natural pedestrian traffic. Here we had around 12-14 pedestrians come into the monitored area instead of just our actors. We wished to test this scenario because it is well known that as the number of foreground objects increase, the reliability of the visual hull algorithm decreases significantly when using a small number of cameras. To accomplish this we deployed our cameras near a busy campus building that has three large columns at one end of the edifice and again captured half-resolution video for about six minutes resulting in just over 11000 frames. We refer to this experiment as the ARCHITECTURE experiment, and the three views can be seen in Fig. 3, again with the results depicted below the views.

The third experiment again demonstrates how the approach, over time, learns occluded areas and tends not to generate many false positives. In the first view we can see that the occlusions are recovered very accurately, especially for having so few cameras. The second and third views are also show much of the occluded areas recovered, with some false positives. Similar to the BIG ZERO experiment, background subtraction played a part in some of the false negatives we see in these images. Because the first image has a large dark window area in the background, and many people are wearing fairly dark clothes, we have quite a few misses in background subtraction. This is the reason there is a gap in the center column in the second view. Similarly, there are misses in the third view on the rightmost column, where we get the top area of the column (where a pedestrian's head would be) but the bottom area is missed.

Even with these errors, the method delivers some very compelling results for and outdoor scenario with quite a few pedestrians and only three wide-baseline cameras from which to extract information.

4.4. Application to Tracking

There are a number of applications for an occlusion model like the one proposed here. For instance, one could use this to ignore certain areas in an image when solving the object detection problem. Or as in [8], the occlusion model could be used to recover visual hulls from partially occluded silhouette images. We actually already solved this problem with our MRF as we can reconstruct areas that are

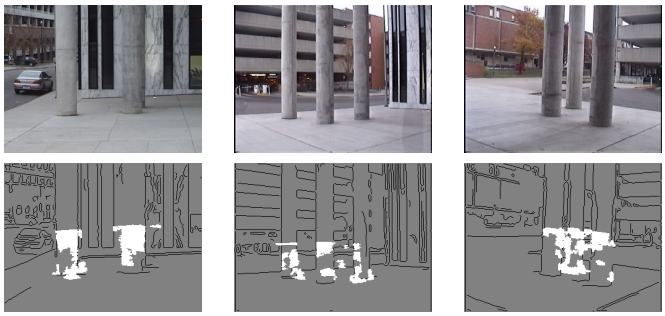

Figure 3. Top row: the layout of the ARCHITECTURE experiment. Bottom row: results on this video sequence.

even fully occluded in one view because of our novel energy function.

We would also like to show how a 3D model like this one could be integrated into a 3D tracking algorithm. For this purpose, we implemented a version of the tracker in [20]. This tracker is a 3D version of the mean shift tracker popularized by [6]. We will discuss the parts of the tracker modified for occlusion reasoning here, but refer the reader to [20] for further details.

To extend mean shift to 3D, [20] uses a feature fusion approach by combining information from all cameras in the 3D space. They extend the mean shift equation to by sampling 3D points, and weighting these 3D points by a kernel which takes into account these fused features. We further extend the method by adding a factor to this weighting term:

$$\hat{q}_u = C \sum_{i=1}^{N} \sum_{\mathbf{V}_j \in \mathcal{N}(0)} \varphi_i(\mathbf{V}_j, u) k(\mathbf{V}_j) w_j^i \tag{14}$$

$$\hat{p}_u(\mathbf{X}) = D \sum_{i=1}^{N} \sum_{\mathbf{V}_j \in \mathcal{N}(\mathbf{X})} \varphi_i(\mathbf{V}_j, u) k(\mathbf{V}_j - \mathbf{X}) w_j^i \tag{15}$$

where $\mathbf{V}_j$ is voxel $j, \mathcal{N}(\mathbf{X})$ denotes the neighborhood of $\mathbf{X}$, $\varphi_i(\cdot, u)$ evaluates to 1 only if the voxel argument projection in view i corresponds to color bin u in the histogram, and C and D are normalization constants. These are the standard equations from the original mean shift paper and the 3D version. There is only one change. It is the function w_j^i:

$$w_j^i = 1 - P(\mathcal{O}_j^i) \tag{16}$$

The standard feature fusion approach adds all histograms with equal weight, which is what makes it robust to rotations of the object in space. However, when an object becomes occluded, especially in more than one view, feature reliability can become an issue. However, in our algorithm, when we know a feature is occluded in a particular view, we weight it much lower than the original algorithm, allowing the feature fusion approach to ignore faulty data. That is exactly what this weight term w_j^i accomplishes.

We show qualitative results in Fig. 4. Due to space reasons, we include only 2 sequences, showing only the occluded camera view. The original tracker is depicted as the yellow dot inside a red circle, and the new tracker is depicted as the green dot inside the blue circle. From this figure, we can see where the occlusion model can significantly increase the performance under full occlusion in a view, and justifies the motivation of the approach by showing its applicability in a real world scenario. In the first frame, the trackers are seen lined up with one another, but as time goes forward, we see that when faced with full occlusion in one of the views, tracker performance can suffer. However, when using our occlusion model, the tracker can overcome the occlusion problem and recover by following the tracked object all the way behind the large pillar by ignoring the corrupted information from the faulty view.

5. Conclusion and Future Work

In this paper we presented a method for modeling 3D occluded areas that is effective even when the number of cameras is small the environmental conditions are imperfect. The paper's main contributions are an iterative method to recover these occluded areas, which itself includes a novel energy function specifically designed to recover these 3D regions. Further, we have discussed two applications for which our method could be used to improve results, the first being 3D reconstruction under occlusion, and the second being a simple extension of a 3D tracker which fuses features from multiple cameras. Experimental results were provided in all cases to support our method.

In the future we would like to extend our method to recover 3D occlusions themselves by growing our occluded regions in images, and then reconstructing these regions. We would also like to integrate the probabilistic model into other trackers on a deeper level, and see how much results can be improved in both single camera and multi-camera tracking. We would also like to integrate a dynamic occlusion reasoner in with this static model, resulting in a tracking system that is robust to both types of occlusions.

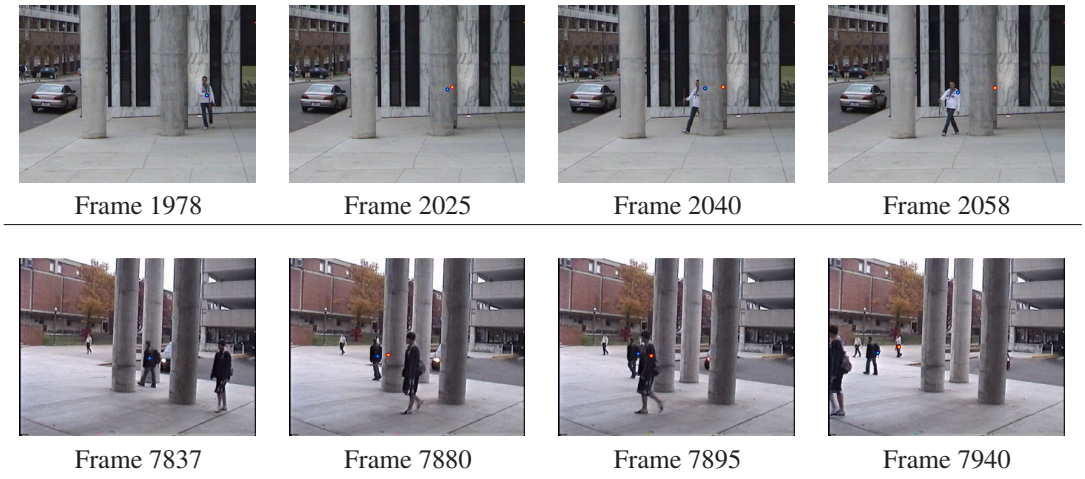

Figure 4. Tracker performance from two sequences in the ARCHITECTURE dataset.

Acknowledgments

We would like to thank Ambrish Tyagi and Vinay Sharma for many stimulating discussions and feedback during the production of this manuscript.

References

[1] N. Apostoloff and A. Fitzgibbon. Learning spatiotemporal T-junctions for occlusion detection. In *Proc. Comp. Vis. and Pattern Rec.*, pages 553–559, 2005.

[2] I. Biederman. Recognition-by-components: A theory of human image understanding. *Pyschological Review*, pages 115–147, 1987.

[3] A. Broadhurst and R. Cipolla. The applications of uncalibrated occlusion junctions. In *Proc. Brit. Mach. Vis. Conf.*, pages 245–254, 1999.

[4] G. Cheung, S. Baker, and T. Kanade. Visual hull alignment and refinement across time: A 3d reconstruction algorithm combining shape-from-silhouette with stereo. In *Proc. Comp. Vis. and Pattern Rec.*, 2003.

[5] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Patt. Analy. and Mach. Intell.*, may 2002.

[6] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. Patt. Analy. and Mach. Intell.*, may 2003.

[7] S. Dockstader and A. M. Tekalp. Multiple camera fusion for multi-object tracking. In *Proc. of the IEEE Wkshp. on Multi-Object Tracking*, 2001.

[8] L. Guan et al. Visual hull construction in the presence of partial occlusion. In *3rd Int'l. Symp. on 3D Data Proc., Vis., and Transmission*, 2006.

[9] L. Guan, J.-S. Franco, and M. Pollefeys. 3D occlusion inference from silhouette cues. In *Proc. Comp. Vis. and Pattern Rec.*, 2007.

[10] Y. Huang and I. Essa. Tracking multiple objects through occlusions. In *Proc. Comp. Vis. and Pattern Rec.*, 2005.

[11] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Trans. Patt. Analy. and Mach. Intell.*, February 2004.

[12] A. Laurentini. How far 3d shapes can be understood from 2d silhouettes. *IEEE Trans. Patt. Analy. and Mach. Intell.*, 17(2):188–195, 1995.

[13] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Patt. Analy. and Mach. Intell.*, July 1990.

[14] M. Pollefeys et al. Visual modeling with a hand-held camera. *Int. J. of Comp. Vis.*, 59(3):207–232, 2004.

[15] A. Senior et al. Appearance models for occlusion handling. In *Proc. Int. Wkshp. on Perf. Eval. of Tracking and Surveillance*, 2001.

[16] V. Sharma and J. W. Davis. Integrating appearance and motion cues for simultaneous detection and segmentation of pedestrians. In *Proc. Int. Conf. Comp. Vis.*, 2007.

[17] Y. Sheikh and M. Shah. Bayesian object detection in dynamic scenes. In *Proc. Comp. Vis. and Pattern Rec.*, pages 74–79, 2005.

[18] J. G. Siek, L.-Q. Lee, and A. Lumsdaine. *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley, Reading, MA, 2001.

[19] D. Snow, P. Viola, and R. Zabih. Exact voxel occupancy with graph cuts. In *Proc. Comp. Vis. and Pattern Rec.*, pages 345–352, 2000.

[20] A. Tyagi et al. Fusion of multiple camera views for kernel-based 3D tracking. In *Proc. Wkshp. Motion and Video Computing*, February 2007.

[21] Y. Zhou and H. Tao. A background layer model for object tracking through occlusion. In *Proc. Int. Conf. Comp. Vis.*, 2003.