Int J Comput Vis (2011) 95:240–264
DOI 10.1007/s11263-011-0446-y

Recovery and Reasoning About Occlusions in 3D Using Few Cameras with Applications to 3D Tracking

Mark Keck · James W. Davis

Received: 30 July 2009 / Accepted: 4 April 2011 / Published online: 22 April 2011

Abstract In this work we propose algorithms to learn the locations of static occlusions and reason about both static and dynamic occlusion scenarios in multi-camera scenes for 3D surveillance (*e.g.*, reconstruction, tracking). We will show that this leads to a computer system which is able to more effectively track (follow) objects in video when they are obstructed from some of the views. Because of the nature of the application area, our algorithm will be under the constraints of using few cameras (no more than 3) that are configured wide-baseline. Our algorithm consists of a learning phase, where a 3D probabilistic model of occlusions is estimated per-voxel, per-view over time via an iterative framework. In this framework, at each frame the visual hull of each foreground object (person) is computed via a Markov Random Field that integrates the occlusion model. The model is then updated at each frame using this solution, providing an iterative process that can accurately estimate the occlusion model over time and overcome the few-camera constraint. We demonstrate the application of such a model to a number of areas, including visual hull reconstruction, the reconstruction of the occluding structures themselves, and 3D tracking.

Keywords Occlusion Recovery · Tracking · Visual hull · Markov random field

M. Keck (✉)
BAE Systems, Inc., 6 New England Exec. Park, Burlington, MA 01803, USA
e-mail: mark.keck@baesystems.com

J.W. Davis
Dept. of Computer Science and Engineering, Ohio State University, 2015 Neil Ave., Dreese Labs Rm. 491, Columbus, OH 43210, USA
e-mail: jwdavis@cse.ohio-state.edu

1 Introduction

It has been shown in the computer vision literature that occlusion reduces the performance of many vision algorithms (*e.g.*, detection and tracking). Because of this, approaches have been proposed that attempt to either implicitly or explicitly deal with this phenomenon in either of its two forms: *static* occlusions, which are unmoving objects in the scene (*e.g.*, trees), and *dynamic* or inter-object occlusions (*e.g.*, people).

Further, because of the growing popularity of multi-camera surveillance systems, there is a push in the community not only to model these occlusions in the image space, but to have some 3D representation of these occlusions. For instance, in an ideal situation, one would place a large number of cameras in the scene, and perform a full 3D reconstruction via a multi-view stereo algorithm. With the full 3D structure of the scene known, the static occlusion problem can be appropriately handled for the 3D reconstruction and 3D tracking problems.

Unfortunately, many practical scenarios (*e.g.*, surveillance) cannot employ standard multi-view stereo algorithms to estimate occluded areas due to practical constraints. First, surveillance scenarios are usually required to run at, or near, real-time. Most multi-view stereo algorithms take many seconds just to process a single frame/set of frames from a camera. Second, multi-view stereo usually requires many views to get an accurate estimate of the depth of the scene; moreover, most multi-view stereo algorithms require relatively small baselines between these cameras. Real-world scenarios will often be constrained to having few cameras which are spread far apart (*i.e.*, wide-baseline configuration).

In this work we propose an activity-based method that learns the locations of static (unmoving) occluding structures in a scene given that we only have a *small number of*

cameras and that they are *configured wide-baseline*, which is an extension to our previous work in Keck and Davis (2008). The method works in two stages. The first stage learns which areas (voxels) in the scene are likely occluded by computing a probabilistic per-voxel, per-view model with an iterative, recurrent procedure. While this model is able to capture which areas are occluded in each view, it does not give an explicit 3D model of the static occlusions themselves. Therefore, in this extension we include a second stage of the algorithm which combines the per-view models to produce such an explicit model. There are multiple advantages to using this method over standard multi-view stereo algorithms: first, the method is fast relative to most stereo algorithms; second, because the temporal dimension is used to learn from static views, fewer cameras are needed to recover the occluding structures; and finally, the cameras may be configured wide-baseline. Each of these address the practical issues related to surveillance scenarios.

The novel contributions of the framework are (1) a method for learning the occluded areas of a scene even when the number of distinct views are few, (2) a new energy functional for visual hull reconstruction via a Markov Random Field (MRF) that incorporates both spatial and temporal constraints as well as a technique for post-processing these results that significantly improves visual hull results, and (3) a method for combining occlusion models across views to recover the 3D occluding structures. We also point out that the primary application of this framework is to aid in computing the visual hulls of foreground objects and improving the results of tracking these objects when under occlusion, and not creating highly accurate 3D models; thus tracking performance will be the primary metric by which we evaluate our system.

The remainder of this manuscript is organized as follows. We will review related work in Sect. 2 and give a full overview to our system in Sect. 3. In Sect. 4 we will present our iterative occlusion learning algorithm, which is able to adaptively learn the occluded areas in the scene, resulting in a per-voxel, per-view model of occlusion. After the learning algorithm completes, we describe a method by which the per-view models can be combined to recover a 3D representation of the occlusions themselves in Sect. 5. We will test these algorithms on multiple datasets in Sect. 6, and then show an application of these 3D occlusion models to tracking algorithms in Sect. 7. Finally we will give concluding remarks in Sect. 8.

2 Related Work

2.1 Occlusion Handling

Occlusion had long been known to reduce the performance of many vision algorithms. For instance, in object detection in 2D (image space), many algorithms are based on learning a person template and finding all locations in an inspection image that are similar to the template (Dalal and Triggs 2005; Oren et al. 1997; Papageorgiou et al. 1998). When holistic templates are used and parts of the object are not present due to occlusion (*e.g.*, a person's legs are blocked from view), the detection performance tends to degrade quickly. Some recent algorithms have *implicitly* given occlusion support to detection algorithms by combining multiple features intelligently (Davis and Keck 2005; Mohan et al. 2004; Tuzel et al. 2007; Viola et al. 2003; Wu and Nevatia 2005) from the training data. We say that occlusions are implicitly handled here because the response of the classifier is a set of weighted component detectors, and as long as the most important features are found (*e.g.*, head and torso in the case of humans), the classification can still be correct even if other components are not present (occluded).

More recent advances in detection not only localize the object, but also allow for the segmentation of the object as well (Gavrila 2000; Sharma and Davis 2007; Wu and Nevatia 2007). These state-of-the-art methods will have a degradation (specifically shown in Sharma and Davis 2007) in performance as the object of interest is occluded more severely and both detection rates and the quality of the final silhouette will gradually worsen.

Implicitly handling occlusion has also been done for tracking algorithms. We consider the effects of occlusion on two broad classes of algorithms: appearance-based algorithms and data associative algorithms. Appearance-based algorithms are characterized by the building of a candidate model of the appearance of an object in an initial frame, and in successive frames search for the object's location by matching to this model. Data associative trackers on the other hand attempt to find frame-to-frame correspondences of observations. In 2D tracking, appearance-based methods deal implicitly with static occlusion by allowing large jumps in frames, as in Porikli et al. (2006), but have inherent difficulty with dynamic occlusions. Algorithms that are data associative in nature also handle static occlusion implicitly (Dellaert and Thorpe 1997; Dockstader and Tekalp 2001; Huang and Essa 2005; Rosales and Sclaroff 1998; Senior et al. 2001), but are usually more concerned with handling dynamic occlusions properly.

Tracking algorithms in 3D have begun to become popular lately, which also offer some implicit handling of occlusion. Algorithms such as Isard and MacCormick (2001), Mittal and Davis (2002) are early 3D tracking systems which employ 3D human body models to aid in the tracking process, which may provide some robustness to occlusion. Other approaches fuse information from multiple single view trackers and triangulate the results in 3D. In Zhou and Aggarwal (2006), shape features are extracted from blobs in each

camera view and they are associated from frame to frame. The individual tracks from each view are then fused into 3D tracks using an Extended Kalman filter. In Zhang et al. (2005) multiple color mean shift trackers in different views were combined to obtain a 3D output. These trackers, because they fuse information at the *decision* level, tend to suffer decreases in performance any time a single tracker fails. To this end, *feature*-level fusion approaches have been proposed to be more tolerant to faulty information in a single view. Approaches including (Black and Ellis 2006; Keck et al. 2006; Khan and Shah 2006) make assumptions about the scene's planar structure in order to obtain object correspondence across multiple views and finally use Kalman filters or clustering to track objects. Very recently a multi-view kernel-based 3D tracker (Tyagi et al. 2007) was proposed which extracted color features from all cameras and combined them into a single feature for mean-shift tracking directly in 3D. This implicitly handles some feature corruption that may occur due to static and dynamic occlusions.

Some explicit occlusion handling methods exist. In 2D, occlusions have classically been found via T-junctions, popularized by Biederman (1987). Some systems exist that employ T-junctions for occlusion finding (Apostoloff and Fitzgibbon 2005; Broadhurst and Cipolla 1999), but practically these features are difficult to extract. Other image space methods exist (Guan et al. 2006; Zhou and Tao 2003), but more recently the focus has shifted to 3D. Guan et al. (2008) recover static occlusions by maximizing a posterior, $p(\mathcal{O}|\mathcal{I},\mathcal{B})$, the probability of a voxel belonging to an occluding structure given a video sequence and background model and are able to track via the same method. The motivation of their method is similar to that of our work: tracking effectively through occlusion. However, their method still requires many cameras to operate effectively and takes many seconds to process a single frame. The intent of our proposed work is to provide an algorithm that can accurately extract the occlusions in the scene in 3D in a more practical scenario, with few cameras configured with wide-baselines.

2.2 Scene Reconstruction

One way to explicitly model the occlusions in 3D is to use a multi-view stereo method, reviews of which can be found in Seitz et al. (2006), Strecha et al. (2008). The motivation of these methods is to create a high-resolution surface of the objects imaged in multiple distinct views. Many of these methods usually require a very large number of cameras (10 or more) for reasonable results as well as often taking minutes per frame to process, both of which are impractical for surveillance scenarios. These methods also do not formally capture the concept of occlusions that transition from static to dynamic over time (*e.g.*, cars parked in/exiting a parking lot).

However, as we will present a probabilistic, voxel-based method for finding occluding structures, one relevant framework to compare against would be the space carving framework, introduced in Kutulakos and Seitz (2000) as a non-Bayesian model and extended in Broadhurst et al. (2001) to provide a probabilistic interpretation to each voxel. In the original method, a plane sweep is performed in each of the principal directions (x-, y- and z-axes) starting from both the positive and negative ends of the axes (six total sweeps) to determine which voxels are photo-consistent across cameras. The photo-consistent voxels are chosen as the reconstruction of the scene, called the *photo-hull*. In the extension, the authors introduce a Bayesian model for determining the photo-hull. Here the appearance of each voxel is modeled and then used to estimate the probability that a voxel is occupied given the data, similar to our MRF model. The goal is to estimate the probability that a voxel is occupied:

$$P(\exists_j = 1|D) = \frac{P(D|\exists_j = 1)P(\exists_j = 1)}{P(D|\exists_j = 1)P(\exists_j = 1) + P(D|\exists_j = 0)P(\exists_j = 0)} \quad (1)$$

The authors also include occlusion probabilities at each voxel by reasoning along rays. The authors extend the original plane sweep method to deal with these occlusion probabilities, and are able generate more photo-realistic results without some of the issues (*e.g.* local hole carving) from the original algorithm.

Both this method and our method produce a probabilistic model at each voxel for each frame. However, we discuss some of the key differences between our method and that of Broadhurst et al. (2001).

Probabilistic Occupancy vs. Probabilistic Occlusion Both approaches provide probabilistic occupancy models over a voxel space. The method from Broadhurst et al. (2001) computes the probability that each voxel is occupied by some object in the scene. We provide a probabilistic reconstruction method that reconstructs only the *foreground* objects at each frame. However, what is finally learned is a probabilistic model of occlusion, not of occupancy. This is a subtle but important point.

Full Reconstruction vs. Occlusion Reconstruction The previous work provides a method for full reconstruction of any object occupying a scene; that is, any object that appears in view of all cameras can be reconstructed given that it is photoconsistent across the views. In our proposal, we will use the learned probabilistic occlusion model to get a reconstruction of the *occluding* objects in the scene. This is a key difference in the two approaches as our algorithm cannot reconstruct any areas that are not occluding structures,

which is symptomatic of the differing motivations of the two approaches. Our approach is looking for a reconstruction for the sole purpose of aiding in tracking applications (Sect. 7), while the previous approaches are concerned with photo-realistic reconstructions of a scene.

Many Small-baseline Views vs. Few Wide-baseline Views Perhaps the most interesting difference between the two approaches is the set of conditions in which each can operate. We have designed our approach to take advantage of temporal information, using a long sequence of frames to learn what areas in the scene are occluded. Because we fuse information over time, it is not as important for our approach to have as many views, which is what effectively allows us to overcome the few camera constraint. Not only that, we will show in Sect. 6 that our approach actually works best when the cameras are well spread. These again are all artifacts of a system designed to be used in practical surveillance scenarios. On the other hand, multi-view stereo approaches usually do not fuse information across time, as is the case in Broadhurst et al. (2001). Here more views are needed to produce reasonable results. The main reason is that the approach is built on examining the variation of the colors onto which each voxel projects. If there are only three views, for example, carving away voxels becomes more difficult as it is much more likely that the projection of a voxel in three views has low variation in the color space, even if the voxel is not occupied. We will show an example of this behavior in Sect. 6.1 when we compare space carving to our approach for reconstruction on a synthetic dataset. What will be seen is that even when many views are used, there may be voxels which cannot be carved away, simply because the projection of those voxels happens to line up in each view with a very similar color, even though the voxel is not occupied.

Furthermore, the space carving approach assumes that all cameras are on one side of the voxel space. This assumption is what allows the efficient plane-sweep implementation to work; otherwise, reasoning about inter-voxel occlusion becomes computationally infeasible. Though this assumption can be relaxed via the six-plane sweep method from Kutulakos and Seitz (2000) and allow cameras to be on multiple sides of the space, this would partition the voxel space into six non-interacting areas. Thus in each of these six partitions, a minimum of two cameras would be needed to carve any voxels away during a sweep, and even more would be needed in each partition to get a faithful reconstruction. Our approach has no such constraints.

In summary, multi-view stereo approaches like space carving are designed to produce photorealistic reconstructions of a scene viewed by many cameras, and perform better than the proposed method for such purposes. Our method, on the other hand, is designed to work under surveillance conditions, producing reconstruction of *only* the occluding structures given only three cameras at wide-baseline configurations, and provides advantages over multi-view stereo algorithms when used in this capacity.

3 System Overview and Motivation

We present a diagram of our overall system architecture in Fig. 1. From this diagram, one can see that the system begins in the learning procedure that will be described in Sect. 4 (the box labeled "Occlusion Model Learning"). A number of subprocesses occur during the learning algorithm. First, background modeling occurs, where a background model is learned over time, and a set of background likelihoods is produced. These likelihoods are then passed forward to the MRF reconstruction module, producing a foreground reconstruction. This foreground reconstruction is then post-processed to remove shadows, *etc.* Finally, the foreground reconstruction is used to update the occlusion model, and this updated occlusion model feeds back to the reconstruction module for processing the frames at the next timestep. The learning process operates continuously on a video sequence, performing these subprocesses for the set of images at each timestep, building a per-voxel, per-view probabilistic model of occlusion over time. It is this temporal processing that allows the algorithm to identify occluded areas even in the presence of only few cameras.

When processing on a video sequence is complete, the occlusion model learned can either be used by the tracking algorithm or be passed to the "Occluder Reconstruction" routine, which is described in Sect. 5. This procedure will convert the per-voxel, per-view learned model into an actual reconstructed 3D object. It begins by smoothing the learned background images from each camera, and then segmenting them via mean shift clustering. These segments are then matched across views, and matching segments that correspond to regions that have been learned to be occluded are reconstructed. Using the occlusion model as a cue for the reconstruction is another element that allows the algorithm to operate with few cameras. This new 3D model of occlusion, which is no longer per-view and only per-voxel, can also be incorporated by a tracker.

Finally, we will evaluate the performance of our tracking algorithm with and without each of these models. We do so by integrating the occlusion model into both a 3D Kalman filter and a 3D mean shift tracker, the details of which are given in Sect. 7. Note that only one of the two occlusion models (either the probabilistic model from Sect. 4 or the binary model from Sect. 5) is used at a given time.

4 Learning of a Probabilistic Occlusion Model

In this section we introduce the foundation of our algorithm, which is a method to learn which areas in a scene are

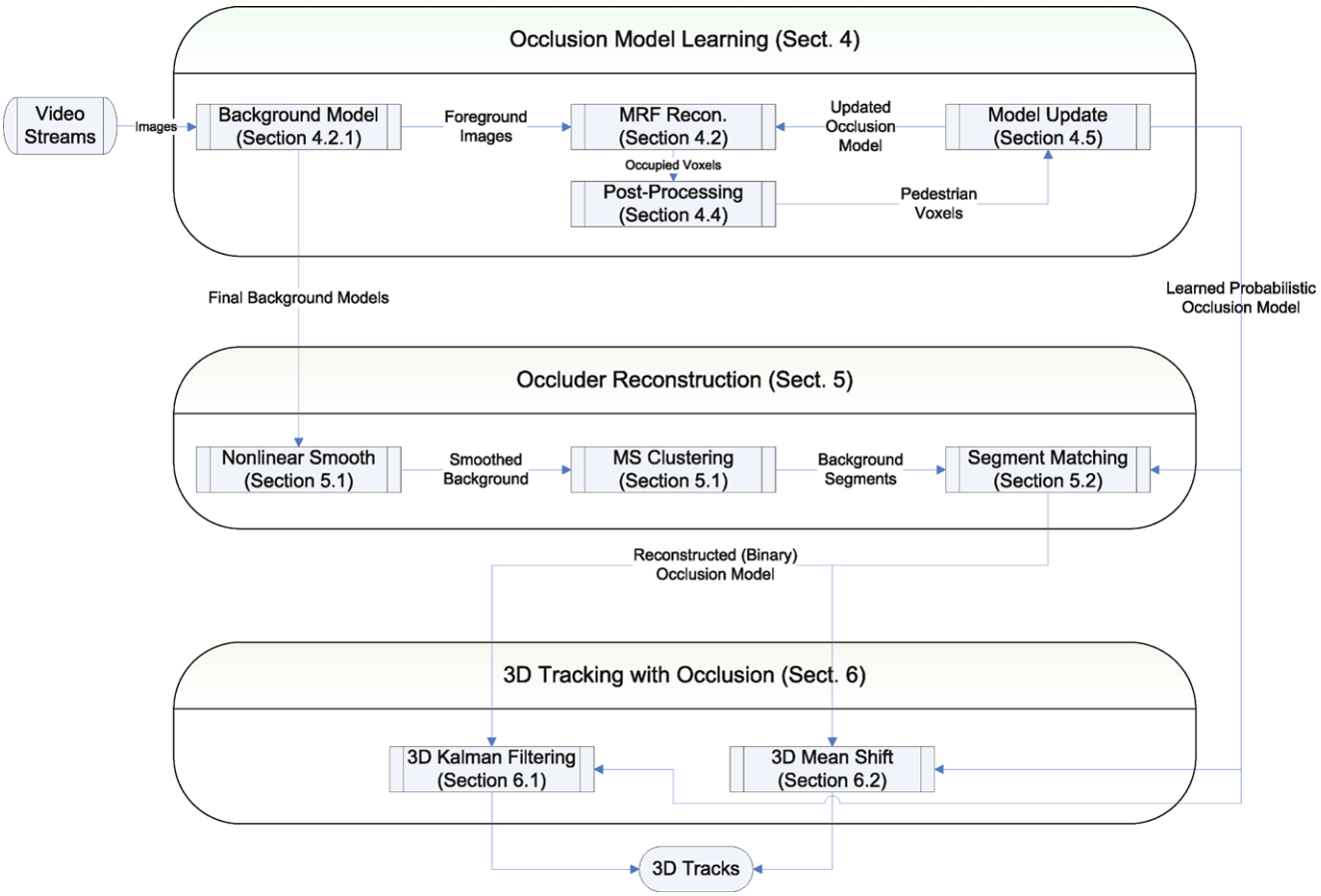


Fig. 1 A diagram of our system, with corresponding section numbers. Processing starts on the upper-left, where images feed into the system. The *top-most large box* encapsulates the occlusion learning procedure, which produces a 3D probabilistic model of occlusion. This model can then be passed to directly to a 3D tracker (*bottom-most box*), or may be processed further (*middle box*) to produce a visual hull of the occluding objects themselves, which may also be leveraged by the tracker

occluded by static structures. The method is able to learn a per-voxel, per-view model of occlusion even when the number of cameras is small. The learning is done by utilizing a 2-phase procedure that is applied to each frame. The first phase (Sect. 4.2) solves the voxel occupancy problem at a given frame with a Markov Random Field (MRF) which utilizes the probabilistic occlusion model. The second phase (Sect. 4.5) employs this solution to update the occlusion model for the following frame. The resulting procedure overcomes the problem of having few cameras by accumulating information over time. We consider this the foundation of the architecture as it will be used as input to the reconstruction method introduced in Sect. 5 as well as the tracking algorithms proposed in Sect. 7.

We will start by presenting the voxel occupancy problem to the reader in Sect. 4.1 and our solution to the problem using an MRF that incorporates constraints for both spatial coherency and temporal consistency. We will then discuss in following subsections how a probabilistic occlusion model can be incorporated directly into the MRF, covering the first phase of the iterative procedure. We will then describe the second phase, discussing how to update the occlusion model at each frame given the voxel occupancy solution.

4.1 The Voxel Occupancy Problem

The voxel occupancy problem is a 3D problem that is analogous to 2D background subtraction. It can be stated as follows: given a scene viewed simultaneously by M calibrated cameras and a 3D voxel lattice defined in that space (denoted $\mathcal{V}$ where $|\mathcal{V}| = N$), determine which voxels are occupied by foreground objects and which are not. This can be interpreted as finding a function $f : \mathcal{V} \rightarrow \{0, 1\}$, where mapping a voxel $\mathbf{v}_j \in \mathcal{V}$ to 0/1 means that $\mathbf{v}_j$ is unoccupied/occupied, respectively. For example, $f(\mathbf{v}_j) = 1$ at time t would indicate that voxel $\mathbf{v}_j$ is occupied by a foreground object (person) at time t. The problem then comes down to determining the optimal function f^* (which can be thought of as a binary labeling of $\mathcal{V}$) for a set of images taken at time t. We will first discuss the most basic solution to this problem, followed by a description of our approach, which

takes advantage of this convenient labeling interpretation by casting the problem into an MRF framework.

The basic approach to solving the voxel occupancy problem is a voting scheme. First, assume that the M calibrated cameras are represented by their projection matrices, which we denote $\mathbf{P}^i$ for $i \in [1, \ldots, M]$. The pixel location in camera i for any voxel $\mathbf{v}_j$ will be referred to with the shorthand notation $\mathbf{p}_j^i = (x_j^i, y_j^i)$.

The basic algorithm begins by computing a foreground image for each of the input views. Any approach that generates a foreground mask can be used, such as Elgammal et al. (2001), Kim et al. (2005), Stauffer and Grimson (1999). We will denote the foreground silhouette image from each camera as $\mathcal{F}^i$. The solution to the voxel occupancy problem (or equivalently the computation of the visual hull of the foreground objects) is then computed by projecting each voxel in $\mathcal{V}$ into each of the foreground images, and tallying a vote count:

$$n_j = \sum_i^M \mathcal{F}^i(\mathbf{p}_j^i) \tag{2}$$

The visual hull is then the set of voxels $\mathcal{T} \subset \mathcal{V}$ such that $n_j = M$. That is, it is the set of voxels that can be "seen" in the foreground of each view.

The major advantage to using such a simple approach is its speed, which could easily achieve real-time performance by harnessing the power of a single GPU. However, some disadvantages do arise with this method. First, at each voxel a difficult threshold decision must be made. That is, if a particular voxel is occluded from any view, then it will never be seen in all M views, even if it is occupied. Although some occlusion handling could be done by reducing the threshold M to smaller number K, one still must choose K, which is likely to vary from voxel to voxel. It would be more convenient to make a fuzzy decision at each voxel. A second disadvantage to this voting scheme is that there is no spatial coherence enforced on the voxels. That is to say, it is likely that neighboring voxels will have the same label due to the solid nature of most objects being considered (*e.g.*, people), and the scheme described above does not enforce this constraint.

We have developed a solution to the voxel occupancy problem via an MRF that addresses these two concerns by pushing the solution into a global energy minimization. This approach also has the advantages of a fast min-cut solution, and will be able to directly incorporate a probabilistic occlusion model.

4.2 MRF Solution

Computing the visual hull via an MRF has been attempted previously in the literature, originally in Snow et al. (2000), where the authors lay out a simple system, but do not develop their approach theoretically from a statistical standpoint. We address that issue in this section, where we model our MRF by combining information from statistical background models in each image. We also prove that our formulation is regular and in the class of $\mathcal{F}^2$ functionals, meaning that it can be solved via a min-cut algorithm.

When solving a labeling problem with an MRF, the typical approach (*e.g.*, Sheikh and Shah 2005) is to maximize the posterior distribution of the labeling given the data, denoted $P(f|\mathcal{D})$, by recognizing that it is proportional to the likelihood of the data times the prior of the labeling. The optimal labeling f^* is defined as the labeling which maximizes this product, but is often transformed into negative log-likelihood space for numerical stability:

$$f^* = \arg\min_f -\ln P(\mathcal{D}|f) - \ln P(f) \tag{3}$$

The heart of the problem then lies in modeling these two distributions, the first being $P(\mathcal{D}|f)$, the likelihood of the data given the labeling, and the second being the prior of the labeling $P(f)$. We will now discuss these two terms in our formulation.

4.2.1 Voxel Likelihood

To estimate the likelihood of the observed data given a particular labeling f, we model both the foreground and background likelihoods in the images for each voxel projection. We create a background model $\mathcal{B}^i$ $(i = 1, \ldots, M)$ for each camera view. In this work, we employ the approach from Stauffer and Grimson (1999) where each pixel is modeled as a mixture of Gaussians. Therefore the likelihood of a voxel being generated by this distribution given that its label is 0 (*i.e.*, $f(\mathbf{v}_j) = 0$) is:

$$P_B(\mathbf{d}_j^i|\mathcal{B}^i) = \max_k \frac{1}{\sqrt{2\pi^3|\Sigma_j^{i_k}|}} \times \exp\left(-\frac{1}{2}(\mathbf{d}_j^i - \mathbf{b}_j^{i_k})^\top \left(\sum_j^{i_k}\right)^{-1}(\mathbf{d}_j^i - \mathbf{b}_j^{i_k})\right) \tag{4}$$

where $\mathbf{d}_j^i$ is an RGB 3-vector in image i onto which voxel j projects, $\mathbf{b}_j^{i_k}$ is the RGB 3-vector to the corresponding pixel in the background model for image i from component k of the mixture model, and similarly $\Sigma_j^{i_k}$ is the covariance of component k for the corresponding pixel in image i.

We model the likelihood that a voxel belongs to the foreground as a uniform distribution, meaning that we assume all colors are equally likely to appear as part of a foreground

object (Sheikh and Shah 2005). We denote this uniform distribution as constant

$$P_F(\mathbf{d}_j^i) = \frac{1}{R \times G \times B} \tag{5}$$

where R, G, and B are the number of possible colors in each of the bands. We will denote this constant as γ. More sophisticated models such as mixtures of different appearance distributions (*e.g.*, Sheikh and Shah 2005; Strecha et al. 2004) could be employed here. However, other works have noted that these more complicated appearance models may increase performance only slightly (Gargallo and Sturm 2005); thus we employed the uniform model for its simplicity.

With these two likelihood functions, we can write the data term as:

$$P(\mathcal{D}|f) = \prod_j^N \prod_i^M P_B(\mathbf{d}_j^i|\mathcal{B}^i)^{1-f_j} P_F(\mathbf{d}_j^i)^{f_j} \tag{6}$$

where $f_j = f(\mathbf{v}_j)$.

4.2.2 Labeling Prior

We must also define a prior on labeling function f. We start by introducing the standard Ising model (Sheikh and Shah 2005; Snow et al. 2000):

$$P(f) = \exp\left[\sum_j \sum_{i \in \mathcal{N}_S(j)} \lambda(f_i \otimes f_j)\right] \tag{7}$$

where $\otimes$ denotes the XNOR operation and λ is a positive constant which is a parameter to the system. The function $\mathcal{N}_S$ defines the spatial neighborhood of a voxel, and in our implementation returns the set of voxels that are directly next to $\mathbf{v}_j$ in each of the x-, y-, and z-directions in lattice $\mathcal{V}$ (*i.e.*, 6-connected). This standard model captures the notion that labelings which are spatially coherent (*i.e.*, neighboring voxels have the same label) are more likely than those that do not.

Unfortunately, when using few cameras, visual hull results at each frame can still be poor. To improve the results within the MRF, we impose an additional constraint that enforces temporal consistency. When processing video at high frame rates (*e.g.*, 30 Hz) it is likely that not only should the labeling of voxels be coherent spatially, but they should also have the same label over short temporal windows. This is because foreground objects will not move far from frame to frame when the video is coming in at a high frame rate and the objects are moving at a normal speed. We can incorporate this into the labeling prior by adding an additional parameter, μ, to the system that encodes how much we believe a voxel labeling should be persistent in time (μ will be high for high frames rates, and near zero when the frame rate is very low)

$$P(f) = \exp \sum_j \left[\sum_{i \in \mathcal{N}_S(j)} \lambda(f_{t_i} \otimes f_{t_j}) + \sum_{u=t-T}^{t+T} \mu(f_{u_j} \otimes f_{t_j}) \right] \tag{8}$$

In this equation we let $f_{t_j} = f_t(\mathbf{v}_j)$ be the labeling of voxel $\mathbf{v}_j$ at time t. The first sum in this term is the same as the basic Ising model, while the second introduces the bias toward temporally consistent labelings, giving a higher prior to those which are consistent along temporal window $[t-T, t+T]$. In all of our experiments, we set $T = 1$.

Given the likelihood and prior terms defined above, the optimal labeling f^* can be computed using the results from Kolmogorov and Zabih (2004). In that work it was proven that any energy functional of binary variables of the form

$$E(x_1, \ldots, x_n) = \sum_i E^1(x_i) + \sum_{i=1}^n \sum_{j=i+1}^n E^2(x_i, x_j) \tag{9}$$

can be minimized by performing a min-cut on a specially constructed graph as long as that energy functional also satisfies the *regularity condition*

$$E^2(0,0) + E^2(1,1) \leq E^2(0,1) + E^2(1,0) \tag{10}$$

In (9), E^1 (often called the data term) denotes a function of a single binary variable at a time, and E^2 denotes a function of two variables at a time, where the double summation ensures that the clique potential between any two nodes i and j is only counted once in the total energy. The important result from Kolmogorov and Zabih (2004) is that no term in the energy function considers more than 2 binary variables at a time. It is clear that (3) is exactly of this form. The first term, the voxel likelihood, considers only one variable at a time. The labeling prior has two independent terms that each consider only two binary variables at a time. It is also trivial to show that this equation satisfies the regularity condition as long as λ and μ are positive. Thus, since no single term in our energy functional considers more than two binary variables at a time, we can directly solve our minimization problem via graph cuts. We thus construct graph $\mathcal{G}$ according to Kolmogorov and Zabih (2004), effectively partitioning $\mathcal{V}$ into two mutually exclusive subsets: $\mathcal{S}$, the unoccupied (label 0) voxels, and $\mathcal{T}$, the occupied (label 1) voxels.

4.3 Integration of the Occlusion Model

Up to this point we have ignored the information we have from an occlusion model; here we integrate the concept of

such a model into the MRF to allow for the reconstruction of areas that are occluded. We begin by defining the occlusion model per-voxel, per-view. We will denote the occlusion model $P(\mathcal{O}_j^i)$ for voxel $\mathbf{v}_j$ in view i. The model will vary between 0 and 1, with $P(\mathcal{O}_j^i) = 0$ meaning that $\mathbf{v}_j$ can without a doubt be seen from view i, and $P(\mathcal{O}_j^i) = 1$ meaning it is certainly occluded from view i.

The estimate of the foreground likelihood can be improved with this model. It is done by realizing that 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*. This notion is formalized in the following way:

$$P_F(\mathbf{d}_j^i|\mathcal{O}^i) = P(\mathcal{O}_j^i)P_B(\mathbf{d}_j^i|\mathcal{B}^i) + \big[1 - P(\mathcal{O}_j^i)\big]\gamma \tag{11}$$

where $P_B(\mathbf{d}_j^i|\mathcal{B}^i)$ is given by (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 something is very likely to be occluded in a given view, it can still be labeled as occupied if the other views believe the voxel is likely to be in the foreground.

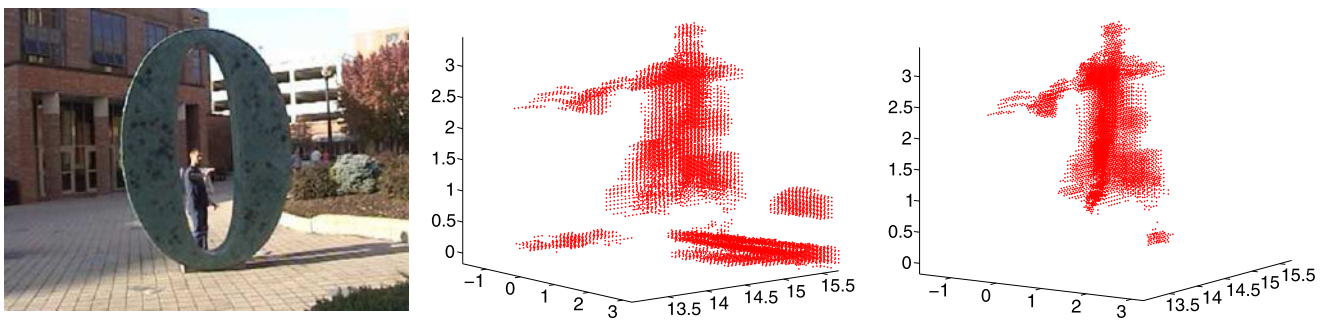
4.4 Shape-based Constraints

Due to the presence of shadows, and the small number of cameras, some unoccupied voxels can still be labeled occupied by the MRF. For instance, in Fig. 2(a) we present one of three views of a scene, and in Fig. 2(b) we show the MRF reconstruction (without temporal constraints). Because of the shadow in front of the large, green occluding structure, which is visible in all three views, a layer of voxels is reconstructed on the ground plane that can affect the learning algorithm. It is important to remove these voxels from the reconstruction. Similarly, due to the noisy nature of the input and difference images, other non-human shapes can be reconstructed like the free-floating blob in the lower right area of the figure.

To relieve some of these issues, we post-process the voxels that are labeled as occupied by the MRF (the set $\mathcal{T}$). We begin by clustering the voxels in the set $\mathcal{T}$ via the mean shift algorithm (Comaniciu and Meer 2002). This results in a set of K clusters $\{\mathcal{C}_i\}_{i=1,\ldots,K}$ that effectively partition the voxels in $\mathcal{T}$ into K disjoint sets. We then examine each cluster, and if there is no voxel in a cluster that is very high off the ground, it is assumed that the cluster belongs to a shadow. Furthermore, we compute the orientation of the ellipse containing the points in the cluster by computing the covariance matrix of the points, and performing an eigenvalue decomposition on the resulting 3×3 matrix, which will give the axes of the 3D ellipsoid containing the points. We remove unexpected shapes by eliminating clusters of voxels that do not have an axis that aligns well with the ground normal. We present an example of the results from this post-processing procedure in Fig. 2(c). The addition of the temporal constraint on the MRF actually removes the large floating blob on the lower right of Fig. 2(b) as this was generated in noise from background subtraction. The shape constraints are able to remove the shadow regions lying on the ground.

4.5 Updating the Occlusion Model

To update the probabilistic occlusion model at each frame, we first compute the foreground silhouette masks in each view with the algorithm from Sheikh and Shah (2005) (note that we previously used Stauffer and Grimson 1999 to *model* the background for the MRF, not to create foreground masks). We chose this method because of its similar energy minimization framework, but again any background subtraction method will suffice. We project each voxel in $\mathcal{T}$ into each of the multiple views. However, some care must be taken in this projection. For instance, it is possible that a second voxel behind $\mathbf{v}_j$ projects to its same location in view i (*i.e.*, they are along the same line of sight). It is also possible that this second voxel is behind an occlusion, but that $\mathbf{v}_j$ is not. Therefore, one can only reason about voxels that are not occluded by any other "on" voxels in each view.



(a) (b) (c)

Fig. 2 (**a**) One of three views of a scene. (**b**) Reconstruction without temporal and shape-based constraints provided. (**c**) Same views reconstructed with our new method

To remedy this, we project the eight corners of each voxel into each view, and compute the convex hull of these eight points via the gift wrapping algorithm (Jarvis 1973). We then at each pixel $\mathbf{p}^i$ keep track the closest voxel from $\mathcal{T}$ for which $\mathbf{p}^i$ was in the projection of its convex hull (*i.e.*, $\mathbf{v}_j$ is the voxel closest to $\mathbf{p}^i$ which is along $\mathbf{p}^i$'s line of sight). The voxels in this set are the only ones that are updated in our system. Note that the same result could be accomplished by doing a plane sweep through the voxels.

For these voxels, we then update the occlusion model by:

$$P(\mathcal{O}_j^i) = \begin{cases} P(\mathcal{O}_j^i) + \alpha & \text{if } \mathcal{F}^i(\mathbf{p}_j^i) = 0 \\ P(\mathcal{O}_j^i) - \alpha & \text{otherwise} \end{cases} \tag{12}$$

where α is the rate of the update. Note that we enforce that the model stays in the range [0, 1]. Although from frame-to-frame, our update will be sensitive to the silhouettes in the foreground masks, errors are smoothed out over time.

5 Reconstruction of Occluding Structures

In Sect. 4 we described an algorithm that is able to learn a per-voxel, per-view model of occlusion. While this model can aid in reconstructing and tracking in occluded areas, it does not provide an explicit 3D model of the occlusions themselves. Such a representation would be more compact (no longer per-view) and would also allow reconstruction/tracking algorithms to reason about occluded areas that were not learned (*i.e.*, never occupied in the learning phase).

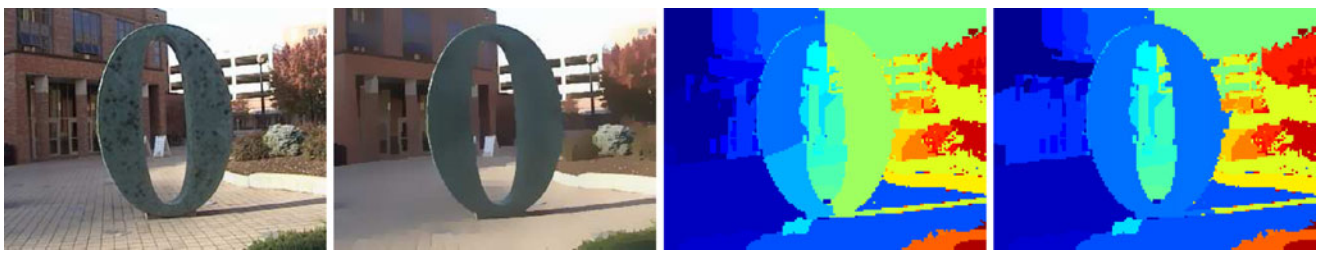
Thus in this section we will introduce a methodology that simultaneously addresses both (1) the issue of incomplete/partial occlusion models for areas that have not been completely learned due to lack of foreground activity in certain views, as well as (2) reconstructing the occluding structures in 3D by combining our per-voxel, per-view learned occlusion models. This new methodology will take advantage of the learned occlusion model (that is, this is performed after learning is complete).

This 3D reconstruction is achieved by first segmenting the input images (Sect. 5.1). This segmentation is then used in a matching framework (Sect. 5.2) which finds segment matches across views using the learned occlusion model as a seed. A 3D representation is generated by computing the visual hull of the segment matches. We wish to stress here that the goal of the reconstruction is not the same as in multi-view stereo, where the objective is to reproduce a 3D object as perfectly as possible, including all concavities. We simply wish to "fill in the gaps" in our occlusion model and reduce the parameter space of the model from $N \times M$ (the number of voxels by the number of cameras) to simply N. This type of model will facilitate the improvement of 3D reconstruction and tracking algorithms which incorporate the occlusion reconstruction while still giving some 3D representation of the scene.

5.1 Segmentation

The process of reconstructing the static occluding structures begins with a segmentation step. The intuition here is that such structures are going to be solid in each image, and that by segmenting the image properly we can match segments across views to reconstruct the object. The segmentation step begins with a nonlinear, isotropic smoothing of the image (Perona and Malik 1990). We use this type of smoother to merge similar colors, while preserving edges in the smoothed images. An example of this smoother on an image from our dataset is shown in Fig. 3(b).

Once the image has been smoothed, it is segmented via mean shift clustering (Comaniciu and Meer 2002). Example results from this operation are shown in Fig. 3(c) where each segment of the mean shift clustering of Fig. 3(b) is given a different color. One can see that mean shift will tend to over-segment large objects in the image, due to the bandwidth of the domain (*i.e.*, (x, y) locations). Because of this, we follow mean shift with a one-pass merging operation where segments that share an edge in the image that are roughly the same color, which is determined by testing the colors with the kernel from mean shift. We show the final result in Fig. 3(d).



(a) (b) (c) (d)

Fig. 3 Clustering results. (**a**) Original image. (**b**) Nonlinear filtering. (**c**) Mean shift clustering. (**d**) Final clustered image after merging (best viewed in color)

5.2 Segment Matching

With a segmentation for all images, the goal is now to find a correspondence between segments in these views. We now present a novel voting scheme to find these matches, which uses the learned occlusion model as a seed in each image.

The process begins by choosing a base image i. The voxels that are highly likely to be occluded from this view are projected into the corresponding segmented image, denoted S_i. For each other view j of the scene, a two-dimensional histogram H^{ij} is constructed. The histogram is subscripted by the segments in each image, for instance $H^{ij}(a, b)$ is the bin corresponding to segment a from image i and segment b from image j, and all bins are initialized to 0.

The weights of each bin are determined by a novel epipolar matching algorithm. For each projected point (x, y) in image i, we determine its segment by indexing $S_i(x, y)$. We will denote this segment s_i. To find a matching segment in image j, we use the well known fact from epipolar geometry that for any two calibrated cameras, we can find for any point in one of the images the line along which its corresponding point must lie in the second image (see Hartley and Zisserman 2004). This method is often used to reduce the search space for correspondences from 2D (the entire image) to 1D (a single line in the image). This is done by computing the fundamental matrix between the two views:

$$\mathbf{F}^{ji} = [\mathbf{P}^i\mathbf{C}^i]_\times \mathbf{P}^i\mathbf{P}^{j\dagger} \tag{13}$$

where † denotes the pseudo-inverse, $\mathbf{C}^i$ is the position of the base camera, and $[\mathbf{a}]_\times$ is the 3×3 matrix that computes the cross product of vector $\mathbf{a}$. Then for any point (x, y) in image i we can compute the corresponding epipolar line $\mathbf{l}^j$ in image j by

$$\mathbf{l}^j = \mathbf{F}^{ji}[x \;\; y \;\; 1]^\top \tag{14}$$

We therefore know that the match for segment s_i in image j must lie along the line $\mathbf{l}^j$. We use this fact to increment our histogram bins. For each segment in $s \in S_j$ that $\mathbf{l}^j$ intersects, the corresponding bin is incremented with the following "vote":

$$H^{ij}(s_i, b) = H^{ij}(s_i, b) + (1 - \|\mathbf{c}_{s_i} - \mathbf{c}_b\|)\left(1 - \frac{A_b}{r \cdot c}\right) \tag{15}$$

Here we denote the color of segment x as $\mathbf{c}_x$, the area of segment x as A_x and the number of rows and columns in image j as r and c, respectively. The colors in this formula are normalized so that the maximum length of any color vector is 1.

This rule formalizes two intuitive notions that we have about matching segments across images. First, matching segments should have similar color, thus the more similar the color, the higher the weight will be. Second, we know that if a segment in an image is very large, then it will naturally be intersected many times based solely on its size. Thus we normalize the weight based on the area of the segment in the second image (smaller regions get higher weights).

Once the histogram is populated with values (*i.e.*, all highly likely to be occluded voxels from image i have been considered), matching segments are chosen by examining each row of histogram H^{ij}. Each row k of the histogram, which we refer to as h_k^{ij}, is first normalized into a probability distribution p_k^{ij}

$$p_k^{ij} = h_k^{ij} / \sum_l h_k^{ij}(l) \tag{16}$$

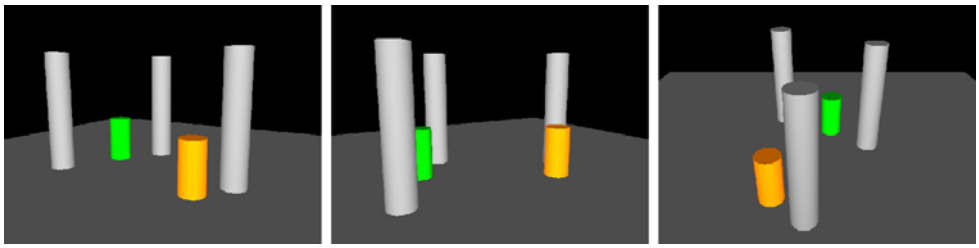
The entropy of this probability distribution is then computed. If the entropy is near 1, we assume that no match exists (because the weight is spread evenly over many of the possible matching segments). Otherwise, the segment with the highest histogram value is chosen as the matching segment. This process results in a set of matching segments across views that correspond to the occluded areas (*i.e.*, segments corresponding to unoccluded areas are not matched/reconstructed).

This process is done from base image i to all images j, and for those images in which a match is found, the reconstruction of the occluding object is found by computing the visual hull of the matching segments. Another image is then chosen as the base image, and the process of seeding this new base image with its occlusion model is repeated, and the visual hull of the matched segments is added to those the previous image base computed. This process repeats until all images have been used as the base image, and the union of all these visual hulls is the final reconstruction of the occluding structures.

6 Occlusion Experiments: Modeling and Reconstruction

In this section we will show through experimentation how well the system we have presented can (1) learn a probabilistic occlusion model and (2) employ that model to reconstruct the occluding structures in the scene. We will test the method on multiple datasets, and provide both quantitative and qualitative results. The datasets vary in content, with both synthetic and real data in indoor and outdoor contexts.

We will first demonstrate the method on a synthetic dataset in Sect. 6.1. The primary purpose of these experiments is to validate our approach from a quantitative standpoint with respect to how well the approach reconstructs foreground objects and models/reconstructs the occlusions in the scene. We will also show how these results vary with the number of cameras. A minimum of 3 cameras is required



(a) (b) (c)

Fig. 4 Synthetic dataset. (**a**)–(**c**) Sample frames from the dataset

because the learning method works by reconstructing objects that are seen in some views but occluded in others. Reliable 3D information requires at least two unoccluded cameras.

We then show the capabilities of the system on a real, indoor dataset with 3 cameras to show that the learning method extends well to real data. Finally, we will show the results of the system on a difficult outdoor dataset.

We also note that we used the values $\lambda = 1.5$ and $\mu = 2.0$ on all datasets, which were selected empirically, and similarly chose the mean shift segmentation bandwidth in the spatial domain (32) and the range domain (16) for all datasets.

6.1 SYNTHETIC Dataset

We created a synthetic dataset observing a space with 3 large pillars that act as occlusions from multiple camera angles. We used OpenGL to create the dataset, which consists of 29 camera views all on a hemisphere pointed toward the scene center (sample frames from the dataset are shown in Fig. 4 (a)–(c)). The cameras capture other smaller cylindrical objects moving around the scene for a duration of 300 frames at a resolution of 640×480. We also used a single Gaussian background model in this experiment, as the background was stationary and we could render it perfectly.

One advantage of using such a dataset is that we can quantitatively compare the results of the learning algorithm while varying the number of cameras used in the procedure. We expect that with more cameras, our algorithm will be more reliable, while with fewer cameras we should encounter a bit more noise.

To quantify how well the system performs at learning occluded areas, we computed an F-measure from a confusion matrix defined over the voxel space. We found those voxels that were highly likely to be occluded (*e.g.*, $P(\mathcal{O}_j^i) > 0.95$) and those that were highly likely to be visible (*e.g.*, $P(\mathcal{O}_j^i) < 0.05$). We will refer to these sets as $\mathcal{A}^i$ and $\mathcal{B}^i$, respectively. We define the true positives to be those voxels from $\mathcal{A}^i$ which are occluded in image i and true negatives as those voxels from $\mathcal{B}^i$ which are visible. Similarly false positives and false negatives can be defined. We can compute the F-measure of this data as a function of the number of cameras, and show the results in Fig. 5.

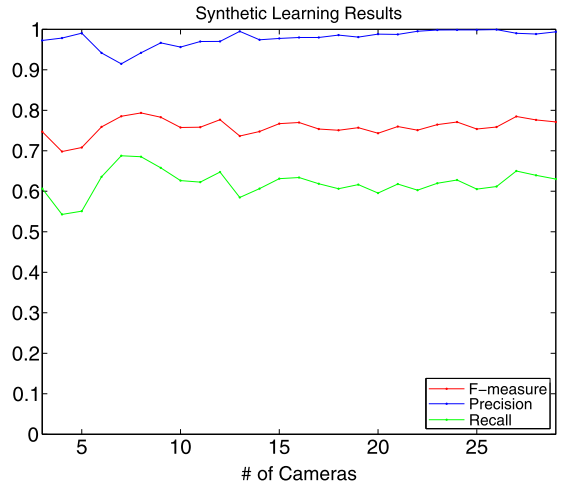


Fig. 5 Results showing the F-measure of the learned occlusion model as a function of the number of cameras used to learn in the SYNTHETIC dataset

The graph in Fig. 5 shows the F-measure, precision and recall. On the x-axis we show the number of cameras, increasing. The main thing that one notices is that the algorithm performance stays fairly constant/stable. The reason that the precision stays so high is due to the conservative nature of the algorithm. The algorithm updates only those voxels it is quite certain are occupied, and $P(\mathcal{O}_j^i)$ will only get near 1 when the algorithm repeatedly believes that a voxel is both occupied and not visible. On the other hand, we also point out that the recall stays consistently lower, near 0.60. This is also related to having an algorithm that has this conservative nature. We will show, however, that a conservative system is much more effective for occlusion reconstruction and tracking.

We show the final results of the learned occlusion model in Fig. 6. We display the results by first plotting the Canny edge map of the mean image from the dataset. Edges are shown in black. All background is shown in gray, and the projection of the highly likely to be occluded voxels ($P(\mathcal{O}_j^i) > 0.95$) in each view are projected in white.

In this figure one can see a reflection of the F-measure results that were presented above: all that is marked as an occlusion truly is an occlusion. In the first row of the figure we show the results of running our system on this data. In

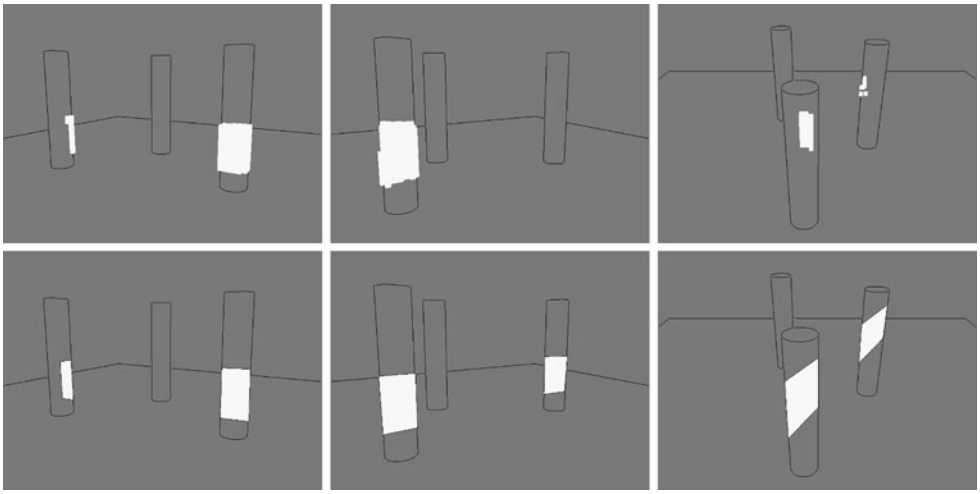


Fig. 6 Synthetic dataset results for learning with cameras 6, 8, and 15. *Top*: Results from the learning algorithm. *Bottom*: Ground truth manual markings of areas where objects were occluded

the second row we show a manual marking approximating all areas that occluded any object in the sequence. The results from the two are similar to one another, and many of the occluded areas that are not projected into the result images are missed because no object occupied that area over the entire sequence (*e.g.*, left most pillar in camera 6). Other areas that are not learned as occlusions are when foreground objects are only visible from one of the views (*i.e.*, they are occluded or off-screen in two views). This is what happens in the right-most pillar of camera 8; when the foreground objects in the scene are occluded by this pillar, they are already off-screen in camera 6, so no learning can occur.

We also present the results of running our matching and reconstruction algorithm from Sect. 5. We first show the results of running the matching algorithm as the number of cameras in the system vary. We report the F-measure of both the reconstruction of the foreground objects for each of the sequences as well as the F-measure for the final reconstruction in Fig. 7(a). A pair of final reconstructions of the occlusions are presented in Fig. 7(b) and (c). In Fig. 7(b), all 29 cameras from the dataset were used in the reconstruction, and in Fig. 7(c), only cameras 6, 8, and 15 were used. We would like to note that because the visual hull algorithm *overestimates* the true object, there is no possible way for the algorithm to achieve perfect results (*i.e.*, F-measure of 1) with a finite number of cameras.

One notices from the graph that, as expected, as the number of cameras is reduced, the reconstruction of the foreground objects becomes poorer, and when the number of cameras is below 10, the results have a pronounced degradation. Even though these results are somewhat unreliable, the reconstruction of the occluding structures themselves is quite accurate even when the number of cameras is small. This is possible because of the conservative nature of the algorithm. Since only occluded areas are learned in each set of images, and because the matching algorithm is able to accurately match the segments belonging to each structure across views, the poor foreground visual hull results are not a significant factor. While using only 3 cameras naturally introduces some error that does not exist when using all 29 cameras, the results are strikingly similar, showing that our algorithm can recover 3D structure even when there are only few cameras.

However, it is important to consider that the placement and coverage of the cameras also plays a role. Consider the following: camera 1, which is viewing the voxel space from the canonical position down the negative z-axis, is on a camera ring with 11 other cameras that has radius 6 units. These first twelve cameras all point to the center of the space, and are spaced apart by 30° on the ring. Thus the closest cameras to 1 are 2 and 12. These cameras are also on one side of the voxel space. If these three cameras were used as the three cameras to model the scene, not all of the columns would be recovered. This is simply because from those three views, two of the columns have areas behind which no activity occurs. Also, when views are closer together, there are much larger areas of the scene which cannot be viewed by at least two of the three cameras, meaning no foreground reconstruction (and therefore learning) can occur. Finally, when using a small number of views, any missed match (match found only in 2 of 3 views) will have a more pronounced effect.

This prompted an experiment testing the coverage of the cameras. For this experiment, we kept the number of cameras constant (3) but varied which cameras were used. We define coverage as the minimum distance between any 2 of the 3 cameras employed. Therefore if any 2 are close together, we consider that low (poor) coverage. To conduct the experiment, 3 views were randomly selected from the 29 views available and the full algorithm was run on this data just as before. This random selection was performed 100 times.

The results of this experiment are shown in Fig. 8(a). This bar graph shows how the F-measure of the system changes

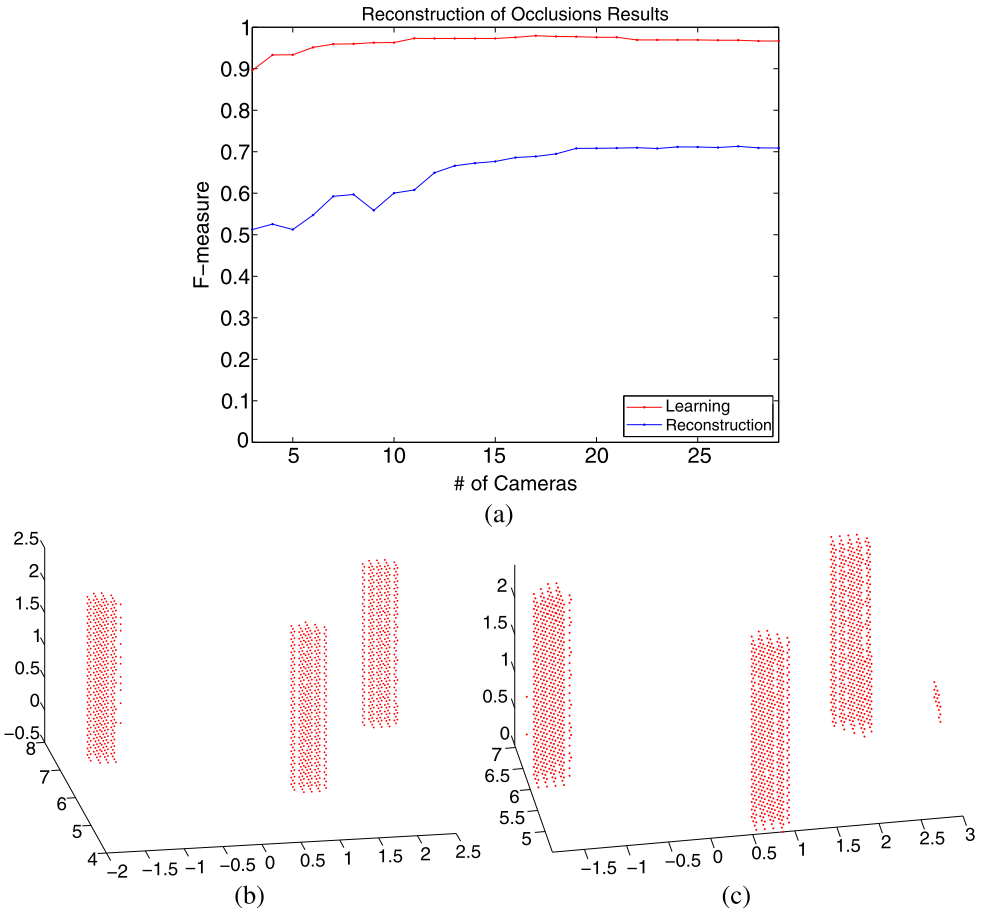


Fig. 7 Reconstruction results for the synthetic dataset. In (**a**) we show the graph of the F-measures of both the foreground reconstructions as well as the occlusions themselves. In (**b**) and (**c**) we show the final reconstructed occlusions using 29 cameras and 3 cameras, respectively

with respect to the coverage of the system. As the cameras lie on a hemisphere with diameter 12, the maximum possible coverage is 12. We partitioned the results into 4 bins, the first with coverage 0–3, the second with 3–6, the third with 6–9, and the final bin with coverage 9–12.

What is evident from the graph is that the results certainly get better as the coverage of the cameras increases. The reason that the final bin is not much higher than the rest is because a degenerate configuration was included here, which was not able to learn the occlusions. While maximizing coverage is desired, one special configuration with high coverage is poor, that of having 2 cameras opposite one another. This occurred in one run where cameras 4 and 10 were chosen randomly together. With the cameras exactly opposite on the hemisphere, when any object foreground object exists between them a very large visual hull is generated. These hulls are naturally removed by our clustering algorithm, and with no activity in the scene, little is learned. Because of this, nothing is reconstructed, since the learning algorithm is the seed. This results in an F-measure of zero for that run. If this were not included, the F-measure for the final bin would be 0.66, a drastic improvement over the previous bins as all other runs belonging to the bin reconstructed at least 2 of the 3 pillars.

Coverage also affects the quality of the final visual hull, even when learning is possible. If the cameras used are close together, then the visual hull reconstruction of the matched segments will be elongated as a match is found in only 2 of the 3 views. We show examples in Fig. 8(b) and (c). In Fig. 8(b), we show the final results when cameras 2, 10, and 14 were used for learning in the scene. Cameras 2 and 14 are very close to one another, with 14 being slightly higher, and right above 2, giving this dataset a coverage value of 3.1682. Because the views of 2 and 14 are so similar, the intersection of the correct matches found between the 3 views produces an elongated version of the actual 3D object (the front-most in this figure). The other two pillars are missed completely (shown here as the darker, rear-most columns). On the other hand, we show in Fig. 8(c) the results from the set of cameras 6, 9, and 18, which have a coverage value of 7.9853. Although not all three columns were correctly reconstructed (the right-rear column, colored darker again, is missed), we do see that the placement of the cameras in this case complements the visual hull algorithm very well, and that the objects that are found have very few false positive voxels around them (some false positives are again inevitable as the visual hull algorithm overestimates the volume of the segmented object).

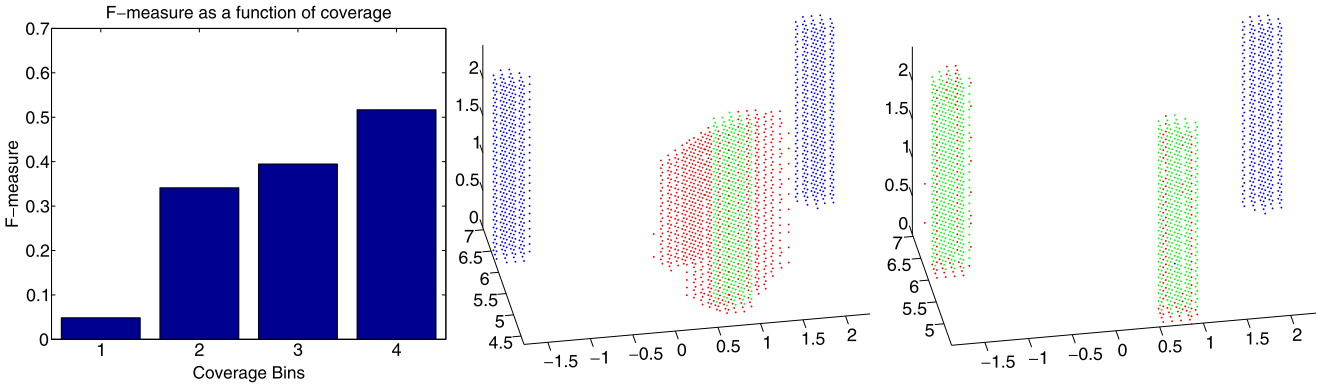


Fig. 8 Effect of performance of the system with respect to the coverage of the cameras. (**a**) Histogram showing F-measure as a function of coverage. (**b**) A reconstruction of the scene with a low-coverage set of cameras. (**c**) A reconstruction of the scene with a medium-coverage set of cameras

This lends great support to our argument that, contrary to most 3D reconstruction algorithms, our algorithm actually performs better with wide baselines between the cameras. Because of this (as well as its activity-driven nature) we believe that the algorithm is well-suited to surveillance scenarios as opposed to image-based rendering approaches.

6.1.1 Comparison to Space Carving

Here we show the results of the probabilistic space carving (Broadhurst et al. 2001) on the SYNTHETIC dataset. Because the approach requires that multiple cameras be one side of the voxel space for good results, we cannot test this approach directly on our real datasets, as in all of them the three cameras are spread around the voxel space. We can, however, test the approach on the SYNTHETIC dataset as we can choose a subset of 3 cameras on one side of the voxel space. Our full implementation performs a six-plane sweep version of the algorithm and automatically partitions the voxels such that they are only reasoned about by at most one camera. This is a fusion of the original version and the probabilistic version. We turned this functionality off, however, when testing with only 3 cameras, as they were all on one side of the voxel space and did not need the added logic. The results of the approach on the SYNTHETIC dataset are shown in Fig. 9, where we plot the photo-hull voxels in the space in their corresponding photo-consistent color. We used as input to the algorithm the static background images from each camera. In Fig. 9(a) we show the results of the space carving algorithm using all cameras from the dataset. As one can see, without any learned occlusion model, this algorithm is able to identify the three occluding structures as well as the floor. The only problem areas for this approach happen around the edges of the voxel space where voxels project to background in every camera and are perfectly photo-consistent as the background is rendered black in all views. As we know that this is not a realistic scene, we believe the approach would normally carve away the background area, and consider the overall performance very good when using all cameras.

On the other hand, in Fig. 9(b) we show the result of running the algorithm with only cameras 1, 2, and 12, all of which are on one side of the voxel space (we cannot use cameras 6, 8, and 15 because they are not). While there is evidence the algorithm is working correctly, as large areas behind the occluding cylinders appear to be carved away with respect to the three views available, one can see that the space carving approach leaves much more clutter this time. The reason for this is that space carving works by modeling each voxel as two competing models: (1) a single Gaussian modeling the color over all cameras, and (2) a set of Gaussians, one for each camera into which the voxel projects. The algorithm attempts to determine which model better explains the data. Many of the voxels rendered as black in Fig. 9(b) project to background areas in each of the three cameras, which is always rendered as black in the imagery. Since these voxels project to each of these views as a black pixel, and therefore a single Gaussian model of the voxel explains the observations perfectly, the voxel is determined to be photo-consistent. As more views are added to the scene, more voxels can be carved away because some view will eventually add information which makes the single Gaussian color model less likely than the multi-Gaussian model for unoccupied voxels. We also show what the space carving approach can do if it knows a particular background color is being used in Fig. 9(c). In this case, we added logic to the algorithm that removed any voxel colored black that had zero variance (*i.e.*, background). With this additional logic most clutter is removed, and it can be seen that the carving algorithm actually does fairly well, keeping only the floor area (which is also a bit more cluttered than the many camera view) and the cylindrical voxels.

This experiment highlights some of the differences between the algorithms discussed in Sect. 2.2. We can see that

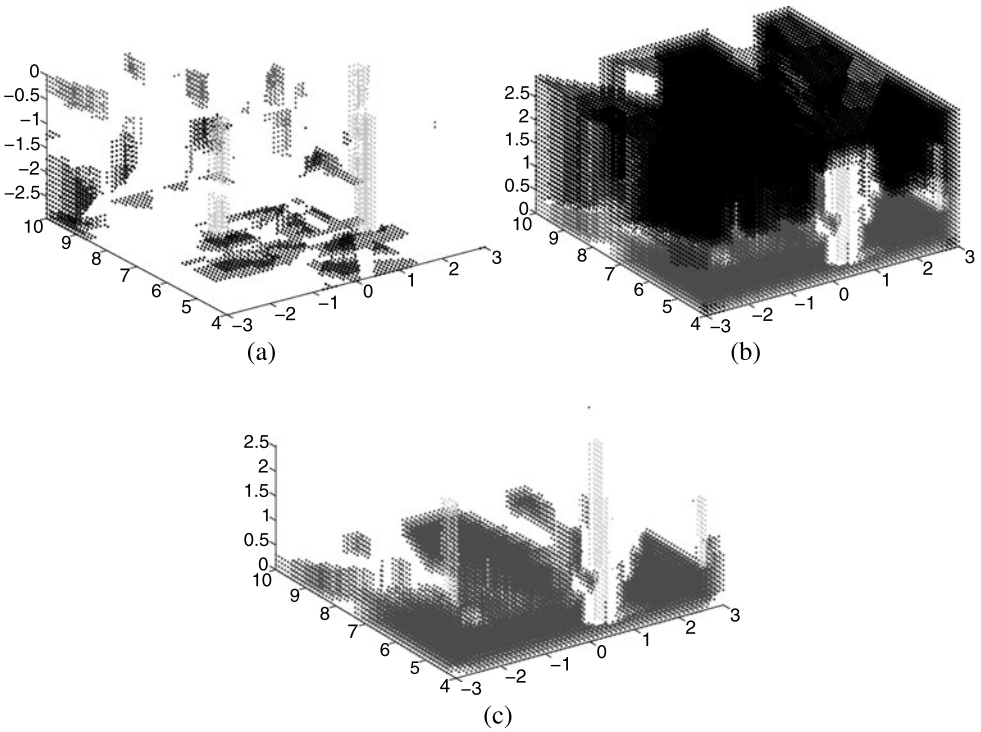


Fig. 9 Results of the space carving approach. (**a**) Results using all cameras. (**b**) Results using only cameras 1, 2, and 12. (**c**) Results from cameras 1, 2, and 12 and removing black background matches

when using all cameras, space carving approaches perform very well and will likely get a more accurate 3D reconstruction than the occlusion reconstruction algorithm proposed. However, when given few cameras the reconstructed scene can become cluttered by voxels that are thought to be photo-consistent due to lack of information. This is one of the advantages to our method: because it uses temporal information to learn which areas of the space are occluded, it can ignore some of these issues automatically, whereas space carving must be supplemented manually. This evidence further supports our previous claim that space carving methods are more well suited to image-based modeling/rendering applications, while our approach is better for learning occluded areas for the purposes of aiding surveillance applications.

6.2 BALL Dataset

The second set of experiments we conducted to validate our learning procedure was done on our BALL dataset, frames of which are shown in Fig. 10. This dataset was captured with 3 Sony Handycams connected to a Matrox Morphis Quad digitizer. We were able to capture the three streams in color at about 20 Hz for around 6 minutes, which resulted in a total of 7630 frames. We created a "lego world" on the floor to create occlusions, and then placed an autonomously moving pet toy (Weazel Ball) into the space to generate the activity necessary, assuming that the object's random movement would be able to cover the space adequately.

We present qualitative results of this experiment in the second row of Fig. 10. We again show a manual marking of the areas where the object rolled behind occlusions in the final row. It is shown that the areas found are truly occluded with few exceptions, and one can see that the second and third rows are very similar. Though some areas are not highlighted (*e.g.*, the top of the closest tower in view 2), this is because the ball is not tall enough to generate activity in those areas.

To validate these results, we again took the F-measure for the learned occlusion model in the same manner as in Sect. 6.1. Using the manually segmented occluding structures in each image an F-measure of 0.8774 was produced, again meaning that the areas we label as positives (definitely occluded) and as negatives (definitely visible) are both accurate and precise.

We used the results in Fig. 10 to seed the segment matching strategy and perform the reconstruction. We show these results in Fig. 11 from multiple angles so that the reader can interpret the 3D reconstruction easily. The reconstruction is shown from an azimuth angle of 18° starting in the top left row all the way to 168° shown in the bottom right. Again we display in Fig. 11 the correct occlusions reconstructed and as well as the false positives from the system. We do not display the false negatives here so that the reader can easily see the precision of the system (we will discuss the false negatives below).

First, one is able to clearly see that the algorithm properly identifies the main structures here (the lego buildings) as occlusions and very accurately matches them. The structure of the inside of the lighter lego building is recreated faithfully.

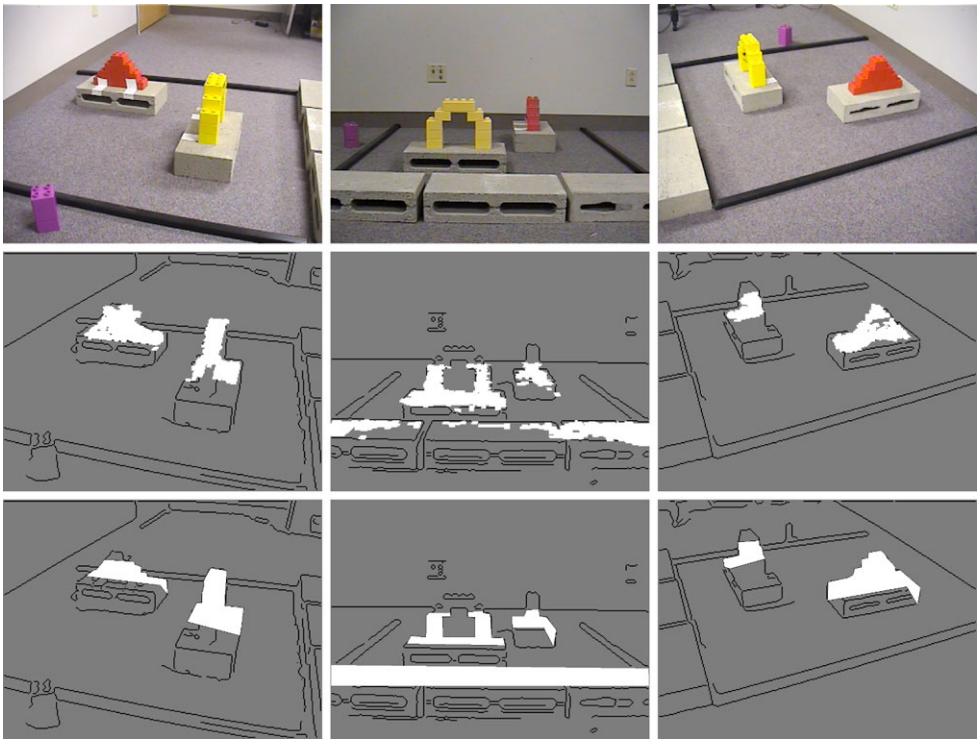


Fig. 10 BALL dataset. *Top*: 3 input frames from the dataset. *Middle*: learned model projected into the three images. *Bottom*: Ground truth manual markings of areas where objects were occluded

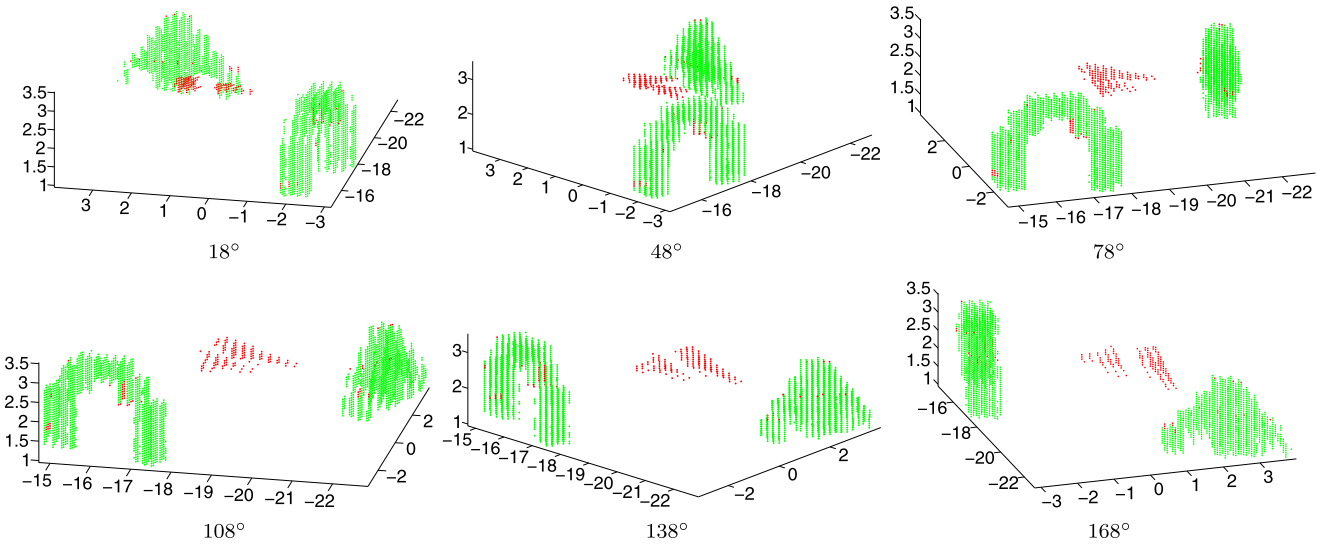


Fig. 11 Multiple reconstruction angles for the BALL dataset

One may also notice that the bottom of the darker building's reconstruction contains two holes that appear to be errors at first glance. However, when one looks at Fig. 10(a), it is obvious that this comes from the segmentation. The red building was taped to the cinder block with white duct tape, which is *correctly* segmented in the image and therefore that area is not matched and reconstructed by the algorithm.

We have also shown the false positives in the plot as well. The false positives shown here are a product of a false match. In the first image, the cinder block underneath the darker building was improperly matched to the back wall in the second image and to a segment on the floor in the third image, creating this floating area in the middle as an occlusion. This is an unavoidable error when the scene cannot be accurately segmented based on joint range-domain data (elements that are similar in color and close to one another are merged into one cluster), and is expected from the algorithm. The important thing, however, is that with such wide

Fig. 12 BIGZERO dataset. *Top*: 3 input frames from the dataset. *Bottom*: Learned occlusion models in each view

angles between the cameras we are still able to find matches and get an accurate reconstruction of the buildings.

We computed the F-measure for this reconstructed data as well, which was a low 0.2464. The reason the F-measure is so much lower on this dataset than the previous dataset is simply because the system was not able to match the cinder blocks across views. First, all of the cinder blocks, as well as the floor, have very similar color, so the matching algorithm threw many out as having no match. Further, some of the cinder blocks were incorrectly segmented and could not be matched correctly. This is obvious when examining the two components to the F-measure formula. On this dataset our precision was 0.93, meaning that what was reconstructed, for the most part, were actual occlusions. However, the recall of the system was 0.14 due to the number of false negatives present for the reasons discussed.

It must be pointed out that again it is more important for our system as a whole to be conservative in what it labels as occluded. We argue that it is much more important for our reconstruction algorithm to be very precise as opposed to being very accurate. This is because the application of the reconstruction is to aid tracking algorithms. If occluded areas are (partially) missed, the tracking performance is not hindered, it merely remains the same. However, if some area is reconstructed as an occlusion that is not, this can adversely affect the tracker, introducing erroneous data. This will be discussed further in Sect. 7.

6.3 BIGZERO Dataset

We introduce the BIGZERO dataset in this section, which is an outdoor dataset with multiple limiting challenges. We employ this dataset to show how our segment matching and occlusion reconstruction component can be used to compensate for poor results expected from the learning algorithm. We captured the dataset from 3 cameras at half-resolution at a rate of 30 Hz, which resulted in 10619 frames. The views from the 3 cameras are shown in the top row of Fig. 12.

This dataset has many issues that will severely limit reconstruction accuracy. It was captured such that multiple areas are occluded from two views (1 and 3). Further, a large specularity is present in view 2 on the occluding structure. With these apparent shortcomings that plague the dataset, we will show that our activity-based model will have difficulty learning, but that our reconstruction will be able to handle these apparent issues, giving results similar to the desired activity-based output.

The results of running the learning algorithm on this dataset are shown in Fig. 12. The top three images show the input frames, and the second row shows the learned probabilistic occlusion model. As expected, the poor nature of the dataset reduced the capability of the learning algorithm to perform well. Many areas of the structure are not learned in views 1 and 3, leaving a very sparse occlusion model as expected because of the inability of the algorithm to learn areas in which no activity occurs. A rather high F-measure of 0.79 is still achieved by the algorithm on this dataset, but largely because very few voxels were even reasoned about.

However, in Fig. 13 we show the resulting occlusion reconstruction on this sequence when seeding the matching algorithm with this sparse occlusion model. Even with a sparsely learned occlusion model and poor lighting conditions, our algorithm is able to reliably segment much of the zero structure in views 1 and 3, and splits it at the specularity in view 2. The robust matching technique is able to find correct matches for the structure in all 3 views (even with some lighting changes) and reconstructs the major part of the zero area. This completion of the inner part of the structure, which the continuous occlusion model does not capture, is very well represented in the reconstruction and therefore the binary occlusion model. In fact, if the specularity had not existed in the scene, our algorithm would have been able to fully reconstruct the zero structure with this

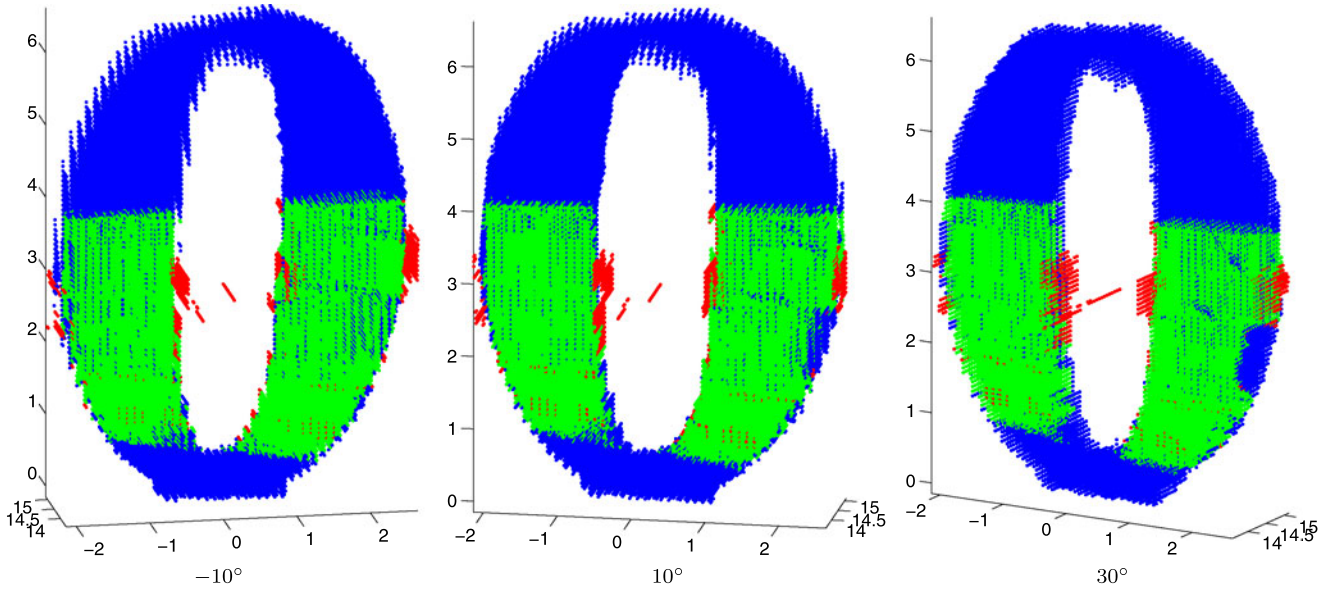


Fig. 13 Multiple reconstruction angles for the BIGZERO dataset

minimal amount of learned information. Overall the results of the reconstruction produced an occlusion model, though not fully segmenting the zero, that is similar to the desired activity-based model for the scene.

In this dataset we were able to mark the ground truth of the occluding structure's shape, and calculate the F-measure to quantify how well our algorithm performed. In this case, the F-measure for this reconstructed structure is 0.6771, due to a low recall and the difficulties presented by view 2, which lead to false negatives. Precision for this dataset was 0.9476, while recall was 0.4798, indicating that very little was reconstructed as an occlusion that was not, while missing the top portion of the structure. However, again we would like to re-iterate that it is much more important for tracking algorithms to have a high precision as opposed to having a high recall rate, which we have again achieved on this difficult dataset.

6.4 ARCHITECTURE Dataset

Finally, in this section the ARCHITECTURE dataset is utilized. Captured near the Architecture building on campus, this dataset presents the biggest challenges to our algorithm. While this outdoor dataset does not have as many illumination issues as BIGZERO, all occluding structures in the scene, as well as the ground, are very similar in color, and many more pedestrians enter/exit the scene during the collection.

We show example frames and results of the learning algorithm in Fig. 14. In all three views, note that the areas that are not occlusions are rarely marked as such. This re-emphasizes the conservative nature of the system, which should only get a very high value in the occlusion model for those areas that truly are occluded. Areas that are not marked as occluded, such as the deepest column in views 2 and 3, cannot easily be modeled since the area behind this column is not visible in both of these views (*i.e.*, it can't be seen by at least two cameras). Thus we cannot model that area.

We calculated the F-measure for our learned model on this scene by marking the 3 large columns manually, and determining which of the voxels we chose indeed were behind occlusions. In this case, our F-measure was 0.7433, which is slightly lower than our indoor and synthetic experiments (as expected), but is still considered fairly high. In this experiment the same phenomenon occurred as before, where our recall was much lower than our precision, which is a consequence of the conservative nature of our system and is viewed as an advantage.

We show reconstruction results from our system in Fig. 15. What we see from the final reconstruction is that one column was missed (the left-most one in the first image), and one has been elongated because a match was found in only two of the three views. Finally the third column has some of the error one would expect around it, but is mostly correct. All of these issues can be attributed to the fact that these views are very difficult to segment and very difficult to match across due to everything looking so similar. However, our results are very encouraging as this scene does not conform to our assumption that occluding objects will be easily differentiated from the background via color.

On this challenging dataset, we are able to get an F-measure on the reconstruction of 0.45, with both precision and recall hovering around this mark due to the false pos-

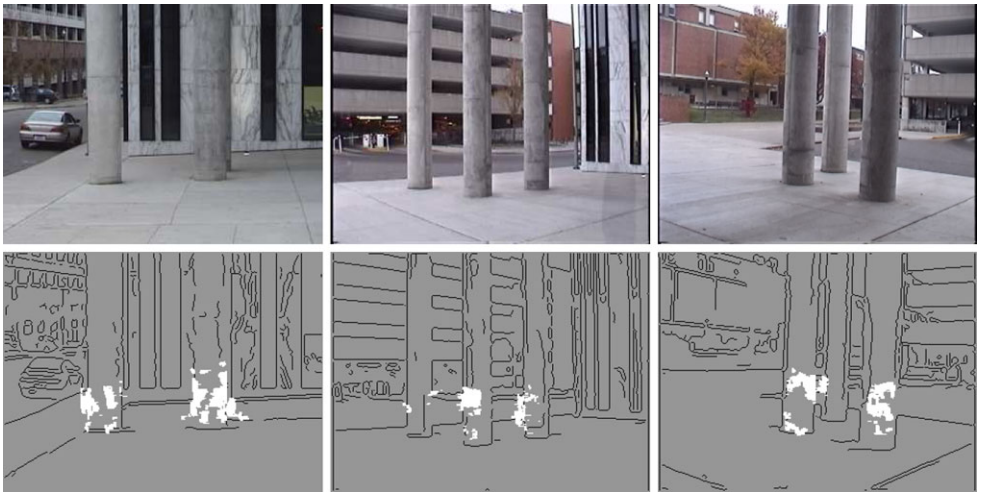


Fig. 14 ARCHITECTURE dataset. *Top*: 3 input frames from the dataset. *Bottom*: learned model projected into the three images

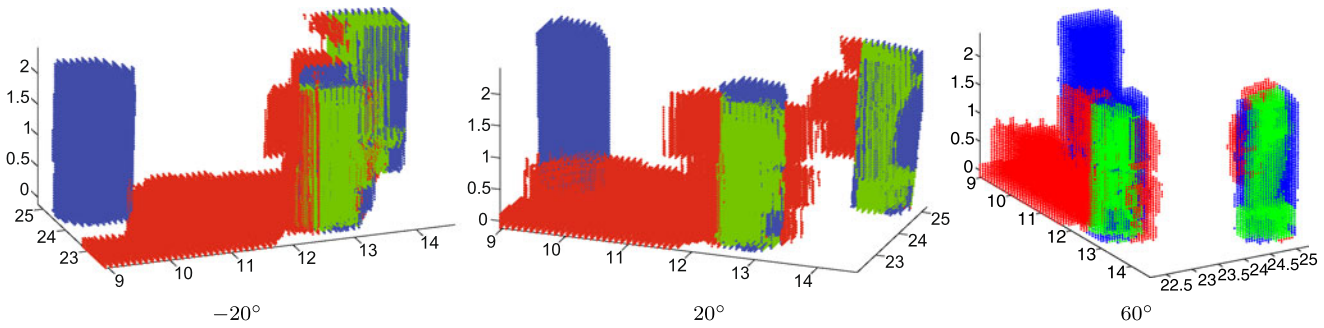


Fig. 15 Multiple reconstruction angles of for the ARCHITECTURE dataset

itives introduced by the two-segment match, and the false negatives from not matching the third column. We consider this dataset an illustration of some of the limitations of our system, and that our algorithm can still perform at some level even in the presence of significant challenges.

7 Application: 3D Tracking with an Occlusion Model

In Sect. 6 it was shown that the algorithms we have proposed for learning the occluded areas in a scene as well as the reconstruction of those occlusions are effective even with few wide-baseline cameras. In this chapter we introduce an application of these models, that of occlusion-robust 3D tracking. This will be accomplished by incorporating the learned occlusion model (or the reconstructed structures) into tracking frameworks so that the tracker can avoid corruption of the tracker state/representation due to occlusion.

We will begin this section by discussing how our representations for occlusion can be used by two popular algorithms for tracking, Kalman filtering (Kalman 1960) and mean shift tracking (Comaniciu et al. 2003; Tyagi et al. 2007) in Sects. 7.1 and 7.2, respectively. We chose to incorporate the occlusion model into these two methods due to their popularity as well as to show that our model can be incorporated into both 3D data associative trackers as well as 3D appearance-based trackers. We present the results of our new occlusion-aware tracking methods *vs.* their baseline counterparts in Sect. 7.4.

7.1 Tracking Through Occlusion via Kalman Filters

Kalman filtering (Kalman 1960), a classic model for discrete linear dynamical systems, assumes a set of noisy observations, $\mathbf{y}_{1:L}$, that are taken at discrete times $t \in [1, L]$, generated by a known process with latent state, and returns as output an estimate of this hidden state at each time step, namely $\hat{\mathbf{x}}_{1:L}$ (the "hat" notation differentiates the estimate of the state from the true, unknown state itself).

The state space form of the Kalman system of equations is shown in (17) and (18)

$$\mathbf{x}_{t+1} = \mathbf{G}\mathbf{x}_t + \mathbf{q}_t \tag{17}$$

$$\mathbf{y}_t = \mathbf{H}\mathbf{x}_t + \mathbf{r}_t \tag{18}$$

where $\mathbf{G}$ and $\mathbf{H}$ are the state transition and observation matrices, respectively, and $\mathbf{q}_t$ and $\mathbf{r}_t$ are the additive Gaussian noise at each state ($\mathbf{q}_t$ is assumed to be distributed zero mean

with covariance $\mathbf{Q}$ and similarly $\mathbf{r}_t$ is distributed zero mean with covariance $\mathbf{R}$).

The implementation of the Kalman filter for video tracking is rather standard in 2D, and here we simply extend it to 3D. We define the state as the 3D position and 3D velocity of the object at time t

$$\mathbf{x}_t = [x \;\; y \;\; z \;\; \dot{x} \;\; \dot{y} \;\; \dot{z}]^\top \tag{19}$$

$\mathbf{G}$ is chosen so that the next state keeps the same velocity, but that the position is changed to add in the velocity. $\mathbf{H}$ is easily defined as well to be the first three rows of $\mathbf{I}_6$ ($\mathbf{I}_n$ is the n-dimensional identity), which means each observation is simply the position of the state at that time. $\mathbf{Q}$ and $\mathbf{R}$ are parameters to the system. We selected these experimentally, and chose the values $\mathbf{Q} = 0.8\mathbf{I}_6$ and $\mathbf{R} = 5\mathbf{I}_3$. These parameters are chosen in such a manner (as opposed to units such as feet/meters) because the 3D reconstruction is metric up to a scale factor, hence the unitless values. With the system identified, the optimal state sequence can be estimated given the standard iterative procedure (see Kalman 1960).

The tracker thus is occlusion-aware as the model $P(\mathcal{O}^i_j)$ is directly integrated into the MRF (the 3D cluster centers of which are the observations to the Kalman filter at each time step). We expect improvements in tracking as the MRF with occlusion knowledge will be able to reconstruct foreground objects that are occluded from some views, while the baseline algorithm will not. This will alleviate errors that arise from missing data because the tracker is provided an occlusion-robust observation at each time step.

7.2 Tracking Through Occlusion via Mean Shift

To show how our occlusion model could be included in an appearance-based tracking method, we employed the 3D extension of the mean shift tracker (Tyagi et al. 2007), where a color histogram is created as the representation of the object in the initial frame, and the location of the object in successive frames is found by getting the best matching candidate location by taking the Bhattacharyya coefficient between the target representation and the representation of the candidates, which is also a color histogram. We refer the reader to Comaniciu et al. (2003) for details on the original tracking algorithm.

In the case of occlusion, mean shift exhibits feature corruption, which means that the color feature extracted from the correct object location will not match the target model well. To alleviate this issue of feature corruption in our 3D tracker, we simply ignore information from sensors that are corrupt. We can model this in the following way:

$$\hat{q}_u = C \sum_{i=1}^{N} \sum_{\mathbf{Y}^* \in \mathcal{S}(\mathbf{0})} R(\mathbf{P}^i \tilde{\mathbf{Y}}^*, u) k(\mathbf{Y}^*) p^i_j \tag{20}$$

$$\hat{p}_u(\mathbf{X}) = D \sum_{i=1}^{N} \sum_{\mathbf{Y} \in \mathcal{S}(\mathbf{X})} R(\mathbf{P}^i \tilde{\mathbf{Y}}, u) k(\mathbf{Y} - \mathbf{X}) p^i_j \tag{21}$$

where $\{\hat{q}_u\}_{u=1:B}$ is the target color model with B bins, and similarly $\{\hat{p}_u(\mathbf{X})\}_{u=1:B}$ is the candidate color model at location $\mathbf{X}$. Because these are assumed to be probability distributions, the coefficients C and D are included to make sure the histograms sum to 1. The function $R(\cdot,\cdot)$ maps a pixel color to its bin u, and the function $\mathcal{S}(\cdot)$ defines the neighborhood of its argument. Finally, the kernel $k(\cdot)$ is the density estimator for these equations, which we choose to be the Epanechnikov kernel (Comaniciu et al. 2003).

The only difference between our equations here and the original 3D mean shift equations given in Tyagi et al. (2007) is the weight p^i_j, which is in the equation to add robustness to occlusion. One straightforward way to define this would be $p^i_j = 1 - P(\mathcal{O}^i_j)$. Thus if the tracked object is moving through an area that is known to be occluded in view i, the features corresponding to the occluded areas of the object are ignored from that view. In some cases, the $P(\mathcal{O}^i_j)$ may not be stable at j because it has been occupied very rarely. In this case, we search along the line of sight between the j and the camera center $\mathbf{C}^i$ of camera i and instead use the maximum value from all voxels in front of j that are stable.

While this model handles static occlusions, dynamic occlusions are not considered. If one object walks in front of another, corruption can still occur. To remedy this problem, we also project all tracked objects into each view, and only update those voxels which do not have another object in front of it. This is known directly from the tracking algorithm, since each tracker consists of a set of 3D locations (lattice) that are assumed to belong to the object. Therefore, the final formula for computing p^i_j is

$$p^i_j = \begin{cases} 0 & \text{if a dynamic object occludes voxel } j \text{ from } i \\ 1 - P(\mathcal{O}^i_j) & \text{otherwise} \end{cases} \tag{22}$$

7.3 3D Tracking with Known Occluding Structures

In the case that no probabilistic occlusion model of the space is available, but some 3D representation is available of the occluding structures (such as that given in Sect. 5), a slightly different model may be used in the MRF to address the problems presented by occlusion.

The idea is to move from weights in the interval $[0, 1]$ to binary weights from the set $\{0, 1\}$. This reflects the concept that either a voxel, from each view, is behind a static occluding structure or it is not. There is no longer a concept of "likely occluded". Thus for each element in the occlusion model, we set the value $P(\mathcal{O}^i_j)$ based on the known occluding structures. This is done by projecting each voxel

$\mathbf{v}_j$ into each view i along with the occluding structures, and determining which one is closer to the camera. If voxel $\mathbf{v}_j$ is closer than any known occlusion in view i, $P(\mathcal{O}_j^i)$ is set to 0 (unoccluded), and otherwise this is set to 1 (occluded).

Specifically, in the case of data associative tracking, where reconstruction is the most important step so that missing data is lost, this is equivalent to changing our foreground model in our MRF to

$$P_F(\mathbf{d}_j^i|\mathcal{O}_j^i) = \begin{cases} P_B(\mathbf{d}_j^i|\mathcal{O}_h^i) & \text{if unoccluded by static structure} \\ \gamma & \text{otherwise} \end{cases} \quad (23)$$

This simple switch allows one to use the same MRF method for reconstruction. Note that any 3D representation that can give depth along lines of sight (*i.e.*, at each pixel) can be used in this approach, and is not limited to the idea proposed in Sect. 5.

Similarly for 3D mean shift, we can alter the weights p_j^i to reflect binary weights instead continuous weights. We do this in exactly the same way described above, where p_j^i is still $1 - P(\mathcal{O}_j^i)$, but the occlusion model is now binary. So if a voxel is occluded by a structure (or another object, which is known through its bandwidth lattice), then p_j^i gets the value 0, and otherwise it gets the value 1. We express this mathematically:

$$p_j^i = \begin{cases} 0 & \text{if an object occludes voxel } j \text{ from } i \\ 1 & \text{otherwise} \end{cases} \quad (24)$$

7.4 Experiments: Tracking

In this section we present the results of the two tracking algorithms described above on our datasets, each of which use only 3 cameras (we again restrict the SYNTHETIC dataset to cameras 6, 8, and 15). Each tracker is run on each dataset under 3 different conditions: the basic condition, where no occlusion model is used to supplement the tracking algorithm; the continuous condition, where the learned occlusion model (based solely on activity) is used to assist the tracker; and the binary condition, where the reconstructed occlusions are used in conjunction with the tracker (called binary as we convert the reconstruction to a binary occlusion model as discussed in Sect. 7.3).

To rate the performance of the tracking algorithms, we marked ground truth in each of the real datasets. In the case of the synthetic data, the true location for the centroid of each actively moving cylinder is known at every frame, giving 300 frames of ground truth in the scene. In the BALL dataset, the object (ball) was manually marked every fifteen frames over the entire sequence, giving 482 frames of 3D ground truth. The BIGZERO sequence has 7 human tracks (which included dynamic occlusion) marked at every 10 frames, and has a total of 142 ground truth 3D locations, and in the ARCHITECTURE sequence 13 tracks were annotated at 10-frame intervals giving a total of 498 ground truth 3D locations.

To determine the error at each frame at which ground truth is marked, the 3D location of both the tracker and its corresponding ground truth location are projected into each of the views being used. The error then is the average pixel error from those views:

$$err = \frac{\sum_i^M \|\mathbf{P}^i \tilde{\mathbf{Y}}_{track} - \mathbf{P}^i \mathbf{Y}_{gt}\|}{M} \quad (25)$$

We choose this error metric as opposed to one in 3D so it can be grounded, whereas errors in 3D are less meaningful since each 3D calibration space has a different scale factor. In the case that the error becomes very large (above some predefined threshold T_{pix}), the tracker will be renominated by the ground truth to its known proper location. One could also use centroid locations of clusters to restart the tracking algorithms, but using ground truth eliminates noise in this phase as most trackers are sensitive to this initialization. The restarting of tracks is another method by which we will gauge its effectiveness, and is necessary to grade the performance of long tracks, as in the BALL dataset. For all datasets and trackers, $T_{pix} = 20$.

7.4.1 3D Kalman Filtering

Our 3D Kalman filter was run on all datasets, with the results are presented Table 1. On the left side the results from the SYNTHETIC dataset are shown for each of the three tracking conditions. At the top, the baseline tracker is shown which reports an error rate of 11.03 pixels on average for the two tracks over the 300 frame sequence. This is compounded by a total of 41 restarts in the sequence, 19 for the first track and 22 for the second track. This happens often because of missing data at many consecutive frames, which is due to the occlusion in the scene. When the Kalman filter is able to reconstruct using the continuous occlusion model, the error rate experiences 33% reduction down to 7.42. Additionally, the error rate is able to improve without as much assistance from the ground the truth, restarting only 11 times, a reduction of about 73%. This is simply because the MRF reconstruction is able to utilize the occlusion model and generate observations even when under partial occlusion. Finally in the binary condition an even bigger improvement is seen by a reduction in pixel error rate to 5.08 (a reduction of 53% from the baseline) as well as even fewer restarts, with each track only needing to be restarted twice. The binary condition is able to improve upon the continuous condition in

Table 1 Results of the 3D Kalman filter on all of the datasets

	Synthetic		Ball		BigZero		Architecture	
	Avg. Err	Res.	Avg. Err	Res.	Avg. Err	Res.	Avg. Err	Res.
Baseline	11.03	41	12.89	70	32.43	72	30.19	283
Continuous	7.42	11	3.64	4	27.56	64	20.67	245
Binary	5.08	4	9.80	53	19.32	44	26.36	191

this case because the learned occlusion model for the Synthetic dataset is sparse (see Fig. 6), but the final reconstruction completely characterizes the scene. Because of this significant increase in the amount of information available, even more 3D observations are generated properly, and the tracker is able to run with many fewer errors.

This table also displays the results of the Kalman filter running on the Ball dataset. The baseline error rate of the ball dataset is 12.89 pixels with 70 total restarts over the sequence. When using the binary condition (bottom row), we see that an improvement is made, but not nearly as significant an improvement as before, only reducing the pixel error 23% to 9.80, with a reduction in restarts by 17. The interesting thing about this data is that the continuous condition is able to achieve an error rate of only 3.64 pixels on the dataset, and over the entire 7600 frame sequence is only restarted 4 times. This is another vast improvement over the baseline. Here the result is transposed from the Synthetic dataset, meaning in that dataset the binary condition was more effective, while here the continuous condition is more effective. In the Synthetic dataset, the scene was better characterized by the final reconstruction than the sparse learned occlusion model, while in the Ball dataset, the cinder blocks below the towers were not reconstructed, but are very well represented by the learned model.

The next two columns of this table present the Kalman filter results on the BigZero dataset. We again see a large reduction in error from the baseline condition to the binary condition (32.43 to 19.32, a 40% reduction). The findings also support the hypothesis that the model which better characterizes the scene (continuous or binary) will have better tracking results. The overall error rate has increased from the Synthetic and Ball datasets, which is expected when moving from controlled indoor to less restricted outdoor settings. This is because of difficulties with background subtraction; there is also more missing data due to clustering outdoors because the shapes of the foreground objects are not as consistent as they are in the previous two datasets. Even then, the occlusion model obviously increases the performance of the tracking algorithm, again keeping a lower error rate with fewer restarts from the ground truth. The final two columns show the same performance measures on the Architecture dataset. Again we see significant error improvements (up to a 32% reduction in the continuous case) over the baseline. The Kalman filter performs rather poorly (in terms of number of restarts) for the baseline case in this sequence due to the fact that so few observations are being reconstructed and passed into the tracker because so much occlusion is occurring. This is mitigated significantly in the binary experiment, with a reduction of almost 100 restarts.

7.4.2 3D Mean Shift Tracking

For 3D mean shift tracking, we selected the parameters for the 3D bandwidth, h_{3d}, and the 3D sampling rate, s_{3d}, experimentally (refer to Tyagi et al. 2007 for a full discussion of these parameters). For each dataset, the 3D bandwidth used was the exact same one as used in the clustering procedure. To select the sampling rate, the baseline mean shift tracker was run on all datasets for all sampling rates between 6 and 20. A sampling rate of $s_{3d} = 16$ was finally chosen as it performed reasonably well on all datasets (minimized the overall error for all datasets).

With these two parameters selected, we performed mean shift tracking with both the learned occlusion model and its binary counterpart on the datasets. These results are presented similarly to those in the previous section in Table 2.

Looking at the table we can see significant improvements over the baseline mean shift tracker in all cases. The baseline tracker in the Synthetic dataset performs reasonably well, getting an error rate of about 9.5 pixels. But because of the occlusion of both tracks by one another as well as by the statically occluding pillars, the baseline case needs to be restarted 23 times over the 300 frame sequence, with one track requiring 8 restarts and the other requiring 15. The second track requires more restarting as it is dynamically occluded while the first is not. The mean shift tracker supplemented with the learned occlusion model does slightly better in this particular case. It is able to allow the tracker a slightly better error rate, but not a significant one (gaining only about one pixel in accuracy). This is again because the learned occlusion model is sparse, as stated previously. Large areas of the occluding columns are missed in all views. However, when the full reconstruction is used in conjunction with the mean shift tracker, we see a significant reduction in error rate as well as the number of restarts needed to keep the tracker performing well. We see a drop in error from about 9.5 pixels per frame/object to 4.68 pixels, a reduction of over 50%. A similar result is seen in the number of restarts necessary,

Table 2 Results of the 3D mean shift tracker on all of the datasets

	Synthetic		Ball		BigZero		Architecture	
	Avg. Err	Res.	Avg. Err	Res.	Avg. Err	Res.	Avg. Err	Res.
Baseline	9.50	23	20.37	143	17.01	43	14.92	187
Continuous	8.61	20	12.84	78	14.56	36	10.88	137
Binary	4.68	11	14.01	94	14.34	33	11.82	80

which drops from 23 to 11. This improvement can be attributed to the vast increase in knowledge of the scene when the full reconstruction is present as opposed to the occlusion model, which in this case is much more sparse.

The second two columns of the table display the results from the indoor Ball dataset. In this case, we again see a marked improvement from the baseline case to the improved tracker, except this time it appears in both cases. Obviously not as accurate as the Synthetic dataset because the color models here are imperfect for mean shift, we see from the baseline tracker an average error rate of 20.37 pixels. This happened because in some cases when the ball became heavily occluded, the tracker got lost, straying far from its location and then registering very large error rates before being restarted because of the number of frames that lapsed between being lost and being restarted (ground truth is marked every 15 frames). Again we see an improvement in tracking in the binary condition, when the full reconstruction is used to supplement the tracker, getting a reduction in error rate of about 6 pixels (a reduction of 31%) as well as a reduction in the number of restarts needed in the sequence from 143 to 94. However, more interestingly we see that the continuous experiment actually performs better than the binary, getting a larger reduction in restarts (from 143 to 78) and a further reduction in pixel error rate to 37%. The reason for this is clear: the reconstruction of the occluding structures misses the cinder blocks in the reconstruction, and therefore the binary condition does not believe they are there. On the other hand, these areas are very well represented by the learned occlusion model, which affords the improvement.

Finally, the right side of the table shows mean shift tracking results on our challenging outdoor datasets. The results from these datasets, on average, show less significant improvements in terms of pixel error than the results than the indoor datasets. Here, in the case of the BigZero dataset, the improved trackers had a pixel error of around 14.5 whereas the original tracker had an error of about 17. Similarly, the error on the Architecture improves from about 15 pixels to about 11 pixels. This modest reduction shows that the occlusion model is able to alleviate some error from the datasets caused by occlusion. And while the number of restarts in the BigZero dataset are fairly similar, with the binary case having 33 restarts, the continuous case having 36, and the baseline case having 43, the improvement in restarts from the Architecture dataset is much more compelling. We see a reduction from 187 restarts to 80 restarts in the binary case. The primary reason for such a strong drop for this dataset is that so much of the background is gray. When the algorithm begins ignoring large parts of the gray scene, it can more easily lock in on the unique color of the object without confusion. In the continuous case, so much less of the scene is found to be occluded that the results show a much less drastic drop.

Overall what we see from the mean shift tracker is that when using the Synthetic and Ball datasets, a dramatic improvement can be made over the baseline tracking algorithm by supplementing it with either the learned occlusion model or the occlusion reconstruction results. These results are compelling because the reduction is significant in both the pixel error as well as the number of restarts needed by the algorithm, meaning that the tracker stays more accurate (on the object) with less outside influence, implying less feature corruption or model drift.

However, when the tracking algorithm is taken outdoors, the improvements on the algorithm are much less significant. We see that for our outdoor datasets the reduction in pixel error is modest (around 2–5 pixels) and that the number of restarts is (usually) much more similar. The reason for this is that the mean shift algorithm exhibits feature corruption for other reasons than occlusion. For instance, in the BigZero sequence, we have very significant changes in lighting conditions in different areas. In view 1, for example, as a tracked object comes out from behind the zero on the right side, it will go through a shadow. This change will affect the mean shift histogram and cause a significant amount of feature corruption which is not handled by the occlusion model. These illumination effects are ubiquitous outdoors, and results in a less pronounced improvement, while synthetic and indoor data (which often exhibit more stable lighting conditions) show significant improvements.

8 Conclusion

In this paper we presented methods for finding occlusions in 3D for scenes with few cameras as well as incorporating these models into tracking methods for improved results. Our occlusion recovery scheme works by finding the visual hull of the foreground objects using an occlusion-robust MRF, a primary contribution of our work. With the visual

hull of the foreground objects computed at each frame, a probabilistic occlusion model is then updated with these results, which then feeds back into the MRF in the following frame. Experiments showed that the learning algorithm is conservative in nature and very precise, meaning that those areas which truly are occluded are the only areas which are detected as such. We also presented a method that uses this occlusion model as a seed to reconstruct occluding structures in 3D. This method, motivated by the need to fill in the gaps of the areas that could not be learned by the occlusion model, was shown to robustly recover the 3D visual hull of the occlusions on multiple indoor and outdoor datasets. Finally, we showed that these two representations could naturally be incorporated into 3D tracking algorithms for improved performance in the face of occlusion, both in terms of pixel error and the number of times that the tracker became lost. The approach results in a practical framework for modeling occlusions in 3D with applications to 3D tracking.

References

Apostoloff, N., & Fitzgibbon, A. (2005). Learning spatiotemporal T-junctions for occlusion detection. In *Proc. comp. vis. and pattern rec.* (pp. 553–559).

Biederman, I. (1987). Recognition-by-components: A theory of human image understanding. *Psychological Review*, pp. 115–147.

Black, J., & Ellis, T. J. (2006). Multi camera image tracking. *Image and Vision Computing*, *24*(11), 1256–1267.

Broadhurst, A., & Cipolla, R. (1999). The applications of uncalibrated occlusion junctions. In *Proc. Brit. mach. vis. conf.* (pp. 245–254).

Broadhurst, A., Drummond, T., & Cipolla, R. (2001). A probabilistic approach to space carving. In *Proc. int. conf. comp. vis.*

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Comaniciu, D., Ramesh, V., & Meer, P. (2003). Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proc. comp. vis. and pattern rec.*

Davis, J., & Keck, M. (2005). A two-stage template approach to person detection in thermal imagery. In *Proc. wkshp. applications of comp. vis.*, Breckenridge, CA, January 2005.

Dellaert, F., & Thorpe, C. (1997). Robust car tracking using Kalman filtering and Bayesian templates. In *Conf. on intel. transportation systems*.

Dockstader, S., & Tekalp, A. M. (2001). Multiple camera fusion for multi-object tracking. In *Proc. of the IEEE wkshp. on multi-object tracking*.

Elgammal, A., Duraiswami, R., & Davis, L. S. (2001). Efficient non-parametric adaptive color modeling using fast gauss transform. In *Proc. comp. vis. and pattern rec.* (pp. 563–570).

Gargallo, P., & Sturm, P. (2005). Bayesian 3D modeling from images using multiple depth maps. In *Proc. comp. vis. and pattern rec.* (pp. 885–891).

Gavrila, D. M. (2000). Pedestrian detection from a moving vehicle. In *Proc. European conf. comp. vis.* (pp. 37–49).

Guan, L., et al. (2006). Visual hull construction in the presence of partial occlusion. In *3rd int'l. symp. on 3D data proc., vis., and transmission*.

Guan, L., Franco, J.-S., & Pollefeys, M. (2008). Multi-object shape estimation and tracking from silhouette cues. In *Proc. comp. vis. and pattern rec.*

Hartley, R. I., & Zisserman, A. (2004). *Multiple view geometry in computer vision* (2nd ed.). Cambridge: Cambridge University Press ISBN: 0521540518.

Huang, Y., & Essa, I. (2005). Tracking multiple objects through occlusions. In *Proc. comp. vis. and pattern rec.*

Isard, M., & MacCormick, J. (2001). BraMBLE: A Bayesian multiple-blob tracker. In *Proc. int. conf. comp. vis.*

Jarvis, R. (1973). On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 18–21.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of ASME, Journal of Basic Engineering*, 35–45.

Keck, M., & Davis, J. (2008). 3D occlusion recovery using few cameras. In *Proc. comp. vis. and pattern rec.*, June 2008.

Keck, M., Davis, J. W., & Tyagi, A. (2006). Tracking mean shift clustered point clouds for 3D surveillance. In *ACM multimedia wkshp. on vis. surveillance and sensor networks*, October 2006.

Khan, S., & Shah, M. (2006). A multiview approach to tracking people in crowded scenes using a planar homography constraint. In *Proc. European conf. comp. vis.*

Kim, K., et al. (2005). Real-time foreground-background segmentation using codebook model. *Real-Time Imaging*, *11*(3), 172–185.

Kolmogorov, V., & Zabih, R. (2004). What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Kutulakos, K., & Seitz, S. M. (2000). A theory of shape by space carving. *International Journal of Computer Vision*, 199–218.

Mittal, A., & Davis, L. (2002). M2Tracker: A multi-view approach to segmenting and tracking people in a cluttered scene using region-based stereo. In *Proc. European conf. comp. vis.* (pp. 18–36).

Mohan, A., Papagoergiou, C., & Poggio, T. (2004). Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Oren, M., et al. (1997). Pedestrian detection using wavelet templates. In *Proc. comp. vis. and pattern rec.*, June 1997.

Papageorgiou, C., Oren, M., & Poggio, T. (1998). A general framework for object detection. In *Proc. int. conf. comp. vis.*, January 1998.

Perona, P., & Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Porikli, F., Tuzel, O., & Meer, P. (2006). Covariance tracking using model update based on lie algebra. In *Proc. comp. vis. and pattern rec.*

Rosales, R., & Sclaroff, S. (1998). Improved tracking of multiple humans with trajectory prediction and occlusion modeling. In *CVPR wkshp. on interpretation of vis. motion*.

Seitz, S., et al. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. comp. vis. and pattern rec.*

Senior, A., et al. (2001). Appearance models for occlusion handling. In *Proc. int. wkshp. on perf. eval. of tracking and surveillance*.

Sharma, V., & Davis, J. W. (2007). Integrating appearance and motion cues for simultaneous detection and segmentation of pedestrians. In *Proc. int. conf. comp. vis.*

Sheikh, Y., & Shah, M. (2005). Bayesian object detection in dynamic scenes. In *Proc. comp. vis. and pattern rec.* (pp. 74–79).

Snow, D., Viola, P., & Zabih, R. (2000). Exact voxel occupancy with graph cuts. In *Proc. comp. vis. and pattern rec.* (pp. 345–352).

Stauffer, C., & Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. In *Proc. comp. vis. and pattern rec.* (pp. 246–252).

Strecha, C., et al. (2008). On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *Proc. comp. vis. and pattern rec.*

Strecha, C., Fransens, R., & Van Gool, L. (2004). Wide-baseline stereo from multiple views: a probabilistic account. In *Proc. comp. vis. and pattern rec.* (pp. 552–559).

Tuzel, O., Porikli, F., & Meer, P. (2007). Human detection via classification on Riemannian manifolds. In *Proc. comp. vis. and pattern rec.*

Tyagi, A., Potamianos, G., Davis, J. W., & Chu, S. (2007). Fusion of multiple camera views for kernel-based 3D tracking. In *Proc. wkshp. motion and video computing*, February 2007.

Viola, P., Jones, M., & Snow, D. (2003). Detecting pedestrians using patterns of motion and appearance. In *Proc. int. conf. comp. vis.*

Wu, B., & Nevatia, R. (2005). Detection of multiple, partially occluded humans in a single image by Bayesian combination of edgelet part detectors. In *Proc. int. conf. comp. vis.*

Wu, B., & Nevatia, R. (2007). Simultaneous object detection and segmentation. In *Proc. comp. vis. and pattern rec.*

Zhang, Z., et al. (2005). A joint system for person tracking and face detection. In *Proc. IEEE int. work. human comp. interaction*, Beijing, China.

Zhou, Q., & Aggarwal, J. K. (2006). Object tracking in an outdoor environment using fusion of features and cameras. *Image and Vision Computing*, *24*(11), 1244–1255.

Zhou, Y., & Tao, H. (2003). A background layer model for object tracking through occlusion. In *Proc. int. conf. comp. vis.*