Depth Layers from Occlusions

Arno Schödl and Irfan Essa
GVU Center / College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280, USA
{schoedl|irfan}@cc.gatech.edu

Abstract

We present a method to extract relative depth information from an uncalibrated monocular video sequence. Our method detects occlusions caused by an object moving in a static scene to infer relative depth relationships between scene parts. Our approach does not rely on any strong assumptions about the object or the scene to aid in this segmentation into layers. In general, the problem of building relative depth relationships from occlusion events is underconstrained, even in the absence of observation noise. A minimum description length algorithm is used to reliably calculate layer opacities and their depth relationships in the absence of hard constraints. Our approach extends previously published approaches that are restricted to work with a certain type of moving object or require strong image edges to allow for an a-priori segmentation of the scene. We also discuss ideas on how to extend our algorithm to make use of a richer set of observations.

1. Introduction

Humans are able to tell scene structure even when visual cues are scarce. Imagine the following experiment: We produce a video of a white glowing ghost floating around randomly in a dark room. In the video, taken from a static viewpoint, the furniture in the room is only visible as dark silhouettes, if the ghost happens to move behind it. A person watching the resulting video will be able to learn about the shape and the spatial arrangement of the furniture, without having ever seen the room fully lit. Our goal is to develop a method to extract similar information automatically.

More specifically, we address the problem of separating a scene into a set of depth-ordered layers from an uncalibrated monocular video sequence. We assume that a single *object* is moving through the static *scene*, and is both occluding scene parts, and is itself occluded by them. We extract over time the region of the image that belongs to the foreground object. These extracted regions form a sequence of *blobs*. The boundary of a blob can be either an *object boundary* itself, or a *scene boundary*, which is a depth discontinuity in the scene causing an occlusion of the object (Figure 1). We do not assume anything special about the scene or the object. In particular the object can be non-rigid, and the scene does not have to be presegmented by image edges. From the sequence of blobs, we compute a set of depth-ordered layers, and their opacities, which can be used for applications like video compositing and surveillance.

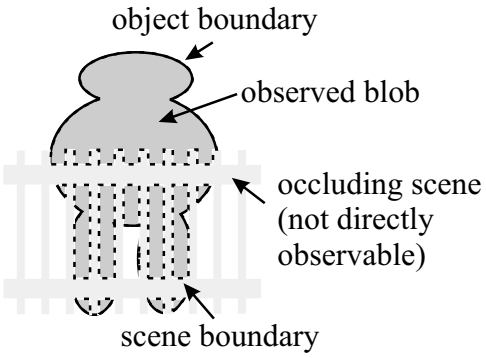

Figure 1: Nomenclature for a simple scene, a person standing behind a fence.

In the course of developing our algorithm, we show that our problem is underconstrained. We use a minimum description length algorithm with a suitable encoding to find a good set of layers and blob-to-layer assignments. We show results on several sequences with a person as the moving object.

2. Related Work

The concept of segmenting a scene into layers is not new to computer vision. Some methods use layers to represent jointly moving regions [9, 2]. In these approaches, optical flow is used to segment a scene into layers of different motion. This approach has been extended to a layer plus height offset representation by Shade *et al.* [5] and in a more re-

cent work by Torr *et al.* [7]. All those techniques use the rigid motion of layers to segment the scene.

Our algorithm on the other hand assumes a static scene, that we segment into depth layers by observing a single non-rigidly moving object. We assume that we can extract this object from the static background using a process known as *background subtraction*. The most naive approach of background subtraction is simple thresholding of the per-pixel color difference of the current image and a stored background image. A lot of research has gone into developing more sophisticated algorithms that allow for lighting and color variations, making background subtraction a well-solved problem in computer vision [8, 3, 10].

Two recent publications use it as an input to identify depth-ordered scene layers [1, 6]. Both assume that the object is at a single depth at a time, rather than spanning multiple depth values. There are two main problems that need to be addressed: (1) It is trivial to say that any area covered by the extracted blob is behind this blob, but the opposite kind of information, which areas are in front of the object, is not as readily available. Areas outside of the blob could be either not part of the object, in which case they do not supply any depth constraint, or they are part of the object, but occluded by scene parts, which would constrain the scene parts to be in front of the object. (2) The depth of the object is usually unknown. Both problems are related, and solving one helps solving the other. Previous research addresses them using heuristics or a-priori scene knowledge.

Stauffer and Grimson [6] determine the depth of the object by assuming that the object is a person moving on a calibrated ground plane. They find the person's head, which is assumed to be always visible, and determine the person's position on the ground plane assuming a constant height.

Brostow and Essa [1] do not determine object depth a-priori, but rely on an edge-based scene segmentation into *regions* of constant depth. They use "motion edges" at those pixels that coincide repeatedly with blob boundaries to help separate neighboring regions that are similarly colored. Then, they use a heuristic that a blob touching a region, but not overlapping it, has been occluded by that region. The region is then labeled as being in front of the object. While this technique often works for large, correctly identified regions, identifying closed regions a priori is brittle, and the heuristic breaks down for small image regions.

3. Our Approach

We observe a static scene in which a moving object is occluding and being occluded by different parts of the scene. Using background subtraction, we observe the visible part of the object at time $t \in 1...T$ as a set of pixels β_t which we refer to as a blob.

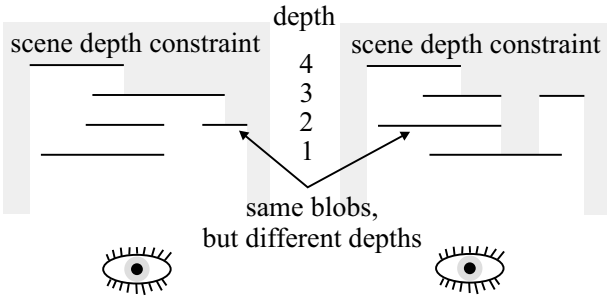

Figure 2: The position of a set of 1D blobs in a 2D world and the inferred scene depths for two different assumed blob depths. The depth assignments are arbitrary and result in different scene constraints.

We assume that the object at any time t is at a single depth $d(t)$. The part of the scene occluded by β_t has a depth larger than $d(t)$. If we know the blob depths $d(t)$, we can assign minimum depths to the scene by applying the depth constraints of the whole sequence to the scene. The depth $\delta(p)$ of a scene pixel p is then bound below:

$$\delta(p) > \max_{\{t|p\in\beta_t\}} d(t). \quad (1)$$

Since we only know which blobs are occluding a scene pixel p, but not which parts of the scene are occluding any blobs, we cannot give an upper bound on the scene depth.

The shapes of the blobs are the result of occlusions, which depend on relative depth differences. Thus, we are in principle restricted to determining relative depth relationships within the scene rather than absolute depths.

3.1. Depth from occlusion is underconstrained

For any given blob sequence $\{\beta_t\}$, the set of constraints given by equation 1 depends on the assumed blob depths $d(t)$. If $d(t)$ is unknown, it is impossible to constrain the depth of the scene. The 2D toy world in Figure 2 shows two different sets of scene constraints for the same observation, assuming different blob depths.

To resolve this ambiguity, we need to obtain some information on the blob depths. We are using the following idea: As stated in Section 1, the blob boundary is either formed by the object boundary or a scene boundary. Scene boundaries will appear as part of the blob boundary if the scene is overlapping the moving object, which can only happen at those pixels where there is a scene depth discontinuity. Object boundary pixels can appear anywhere, depending on the movement of the object. Thus, when a given pixel is repeatedly part of the blob boundary it is more likely to be scene boundary than to be object boundary. In the next

paragraph we will formalize this idea into an algorithm to obtain depth information.

3.2. Depth computation using minimum description length

In order to derive scene depth information in the absence of hard constraints, we use the principle of minimum description length [4, 11]. We discretize the scene depth into a few ($D < 10$) layers and assume that every blob β_t belongs to such a depth layer $d(t) \in \{1, 2, ..., D\}$, where $1, 2, ..., D$ is the order of layers from front to back.

We model the layer at depth d as a transparent set of pixels λ_d. Here, λ_d must contain the union of all blobs that have a depth equal or greater than layer d (we assume the transparent region d is between object depth layers $d-1$ and d).

$$\lambda_d = \cup\{\beta_t | t = 1...T \wedge d(t) \geq d\}. \quad (2)$$

For the minimum description length algorithm, we describe λ_d as the set of pixels on its boundary. Every one of the boundary pixels is assumed to be chosen from the set of all screen pixels. To describe a layer with l boundary pixels, using a screen resolution of s pixels, we need E_M number of bits:

$$E_M \ = \ l \ \log s. \quad (3)$$

The observed blobs β_t are also described in terms of their boundary pixels. Boundary pixels of β_t that coincide with the boundary of the associated layer $\lambda_{d(t)}$ are assumed to be scene boundaries, and are encoded by choosing them out of the layer boundary pixels. This takes $\log l$ number of bits per encoded pixel, where l is the number of boundary pixels of $\lambda_{d(t)}$. Any blob boundary pixel lying in the interior of $\lambda_{d(t)}$ is chosen out of the layer's interior pixels, which for L interior pixels of $\lambda_{d(t)}$ takes $\log L$ bits. Usually, $L \gg l$. Finally, we must encode the decision which group the pixel belongs to, which we do using a per-layer optimal bit allocation. If all the blobs belonging to a layer have a total of b pixels that lie on the layer boundary, and i pixels that lie in the interior, the total number of bits E_D to encode all the blobs belonging to this layer is thus

$$E_D \ = \ b \ \log \frac{b+i}{b} + i \ \log \frac{b+i}{i} + b \log l + i \log L. \quad (4)$$

The total number of bits to describe a layer is $E_M + E_D$, and the total number of bits to describe a scene is the sum of description lengths over all layers. We are looking for the blob-to-layer assignment that minimizes this description length.

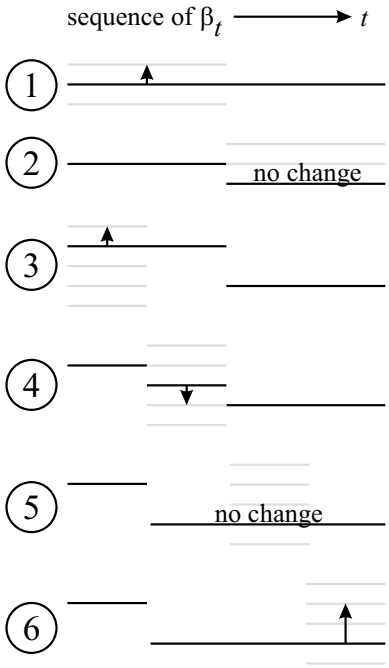

Figure 3: The change of depth assignments for a sequence of blobs during the course of an optimization. The grey lines indicate which assignments are tried, the arrows indicate at each step which assignments are kept.

3.3. Optimizing layer assignments

Obviously there is an exponential number of possible blob-layer assignments, so enumerating all possible assignments is not feasible. Instead, we developed an iterative scheme that computes a sub-optimal, but in practice good blob-layer assignment. We exploit the fact that the moving object usually stays in the same depth layer for some time rather than erratically changing its depth, creating subsequences of blobs that belong to the same layer. So if we adjust the depth of whole subsequences of blobs at once rather than of single blobs, we have a good chance of finding subsequences that we can sort into a coarse layer structure. In practice, we are going from large subsequences to small ones, so that after establishing the coarse structure, we refine the assignments of smaller subsequences or individual blobs.

In particular, we start with an initial assignment of all blobs to a single layer. We then divide the time-ordered sequence of blobs into two halves and try all possible depth assignments of the first half, then the second, then the first again etc., recomputing the description length for every assignment change and keeping the best assignment. After the depth assignments settle, which is the case when we try both halves and none changes its depth, we divide the sequence into quarters and change the depth assignment of

the quarters according to the same scheme. We continue with eighth, sixteenth and so forth, until we change the depth assignments of individual blobs (Figure 3).

Assume we currently use N depth layers (initially, $N = 1$). Then for any subsequence (halves, quarters, ...) whose depth assignment we change there are usually $2N$ possible assignment to try: $N-1$ layers that already exist (minus one for the layer the subsequence currently belongs to), $N-1$ "gaps" between layers and 2 extra depth assignments in front and behind of all other layers. If the best assignment is not one of the already existing layers, a new layer is created. Similarly, if after changing the assignment one of the previously existing layers is empty, it is deleted.

After adjusting the blob depths to the single-blob level, as the final step we are checking whether collapsing any adjacent pair of layers into a single layer improves the result. For the sequences that we present in this paper, this step did not change the result.

4. Results

All our input footage was recorded using an NTSC DV progressive scan camera at 720x480 resolution. Our first test sequence, the *lab* sequence, shows our lab under construction, with a pile of building material lying in the middle. A person enters in the back on the right, walks behind the pile of material to the left and then walks back to the right in front of the pile, but behind a pillar standing in the foreground (Figure 4). A human would probably create three depth layers, from front to back the pillar, the pile of material and the back wall.

To process this sequence, we reduced the resolution to 240x180 to suppress noise, make boundary pixels line up better, and to speed up computation. We extracted the foreground blob by simply thresholding the L_2 color distance. Figure 4 shows the scene and a few blob images. Frequently, shadows have been included into the foreground blob, but they fall on scene parts behind the object, which is acceptable. For the first experiment, to check the validity of our algorithm, we fix the number of layers to two, which results in three different scene regions: The front region, which is never occluded by any blob, the middle region that is occluded by the blobs assigned to the front layer but not by those assigned to the back layer, and the back region that is occluded by both layers of blobs. We expect those three regions to correspond to the pillar, the pile of material and the wall. During the course of the sequence, the person is roughly moving from back to front. We modify the algorithm, so that it assigns the first part of the sequence up to some frame f to the back layer and the rest after frame f to the front layer. The algorithm tests the depth assignments for all values of f to find the assignment of minimum description length. The graph in Figure 5 shows how the the description length changes with f. The minimum is marked with a circle, and the three regions that result from this minimum are shown next to the graph. They correspond very well to the expected segmentation into pillar, material pile and wall.

Now we run our full algorithm, which automatically creates five layers, segmenting the scene into six regions (Figure 6). These six regions correspond roughly to what we expect, but oversegment the scene somewhat. The first region is the same as in the previous two-layer experiment. The second and third contain the material pile, and the last three contain the back wall. Those last three layers clearly show a flaw in our algorithm: When the object is standing still, like on the right side of the scene where the person is closing the door, certain boundary pixels are used repeatedly, which causes the algorithm to create a new layer for this part of the sequence.

In the second sequence (Figure 7) the scene is seen through a window frame. The person enters on the right in front of a table, walks to the left, then back behind the table to the right, where he picks something up. Then he leaves the scene to the left. Again, there are three natural layers, the window frame, the table and the back wall. We reduced the resolution by a factor of two and cropped the image to remove some of the window frame to speed up computation. Our algorithm oversegments the scene again into five layers and six regions. The first two correctly show the window frame and the table, the other four show different parts of the back wall. Here, too, the oversegmentation was caused by the person standing still and picking up something from the floor.

5. Discussion and Future Work

Analysis of both sequences results in oversegmentation when the blob is not moving. One way to overcome this problem would be to adaptively subsample the input sequence, so that consecutive foreground blobs that are almost identical get discarded.

Obviously, the foreground object segmentation is only a small subset of a large number of depth cues in a monocular video sequence of a static scene with a moving object. The most obvious one that we missed, and the one easiest to incorporate into the proposed framework, is accretion and deletion: At scene boundaries, when the object appears or disappears behind a scene part, the texture of the object is moving towards or away from the boundary, while the scene boundary itself remains static. This local movement of texture close to a static boundary could be measured using suitable optical flow techniques and could influence a prior for or against being encoded as object or scene boundary. Overcoming this prior would incur extra encoding costs, so scene interpretations that agree with the

Figure 4: The *lab* sequence. Input scene and blobs extracted from the sequence.

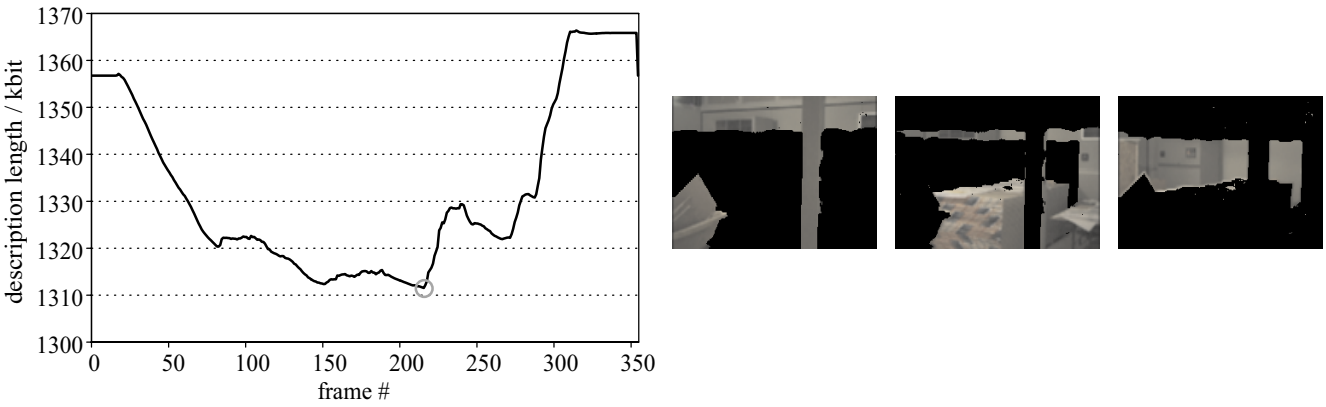

Figure 5: Separation of the *lab* sequence into two layers. The first part of the sequence becomes the back layer, the rest the front layer. The boundary of those two parts is swept over the whole sequence and the description length is plotted. At the minimum description length (circled) we obtain the shown image regions.

Figure 6: Automatic segmentation of the *lab* scene into six regions.

prior are preferred.

Semi-rigid shape constraints for the object can be included in a similar way. We assume the shape of the object at time $t+1$ is a 2D-translated version of the shape at time t, and any observed deviation is an indication for an occlusion, which is again incorporated as a prior.

We also make no use of the correlation between background image edges and scene depth discontinuities. Brostow and Essa [1] relied on this information almost entirely, which made their method brittle. In a more probabilistic approach we would impose a prior on the encoding of λ_d, decreasing the encoding cost for those boundary pixels that coincide with image edges. Unfortunately, λ_d is still restricted to be a union of the blobs, even if the edges strongly indicate to expand a layer. For example, sometimes we may want to include the rest of the uniform blue sky into the backmost layer, although it was never occluded. This expansion could be done in a post-processing step by running a shortest path algorithm over the grid of pixels, weighing every pixel with its prior to be scene boundary.

There are many other important depth cues, that we are not using at present: object shadows and scene lighting, changes in object size or geometric constraints like straight/parallel lines, orthogonalities and planes.

Another shortcoming of our system is that it does not jointly optimize depth assignment and background subtraction result. If the background subtraction system would compute a probability for every pixel being foreground or background instead of a binary result, we could attempt an EM-style optimization, alternating between optimizing the depth assignment and blob coverage. We have not explored any such algorithm so far.

6. Conclusion

In this paper we present a principled approach of inferring depth layers by observing a moving object in a static scene. We use the output of a background subtraction system, which divides the image into static scene regions and

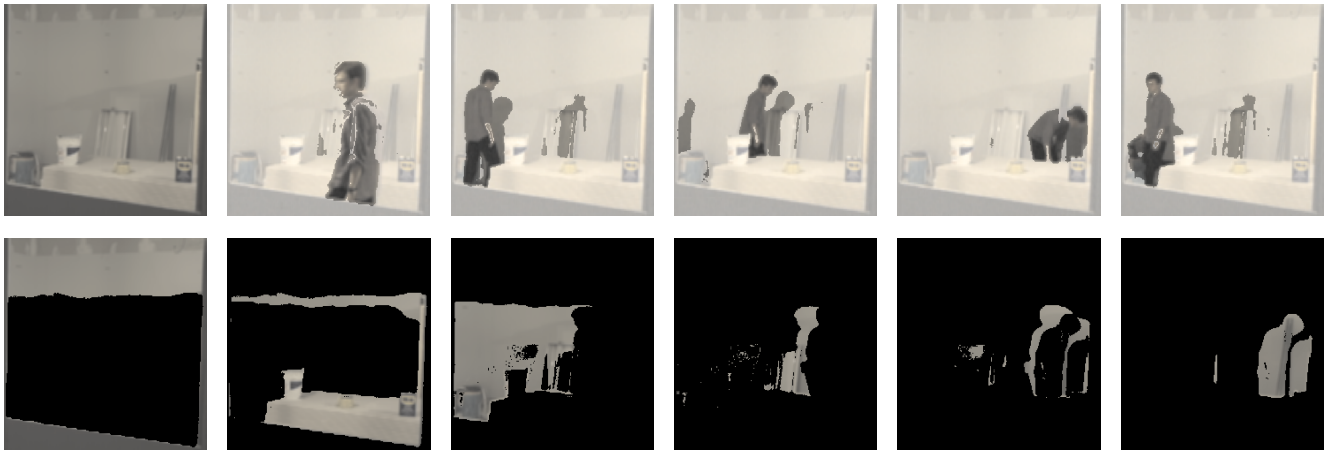

Figure 7: The *construction* sequence. Top row: input scene and blobs extracted from the sequence. Bottom row: automatic segmentation into six regions.

regions covered by the moving object, as the only input to infer depth. The problem is underconstrained in the sense that any possible solution of object-to-depth assignments is consistent with any observation. We make the problem tractable by proposing a suitable encoding of model and data and finding the solution of minimum description length for that encoding. We obtain good depth layers on practical examples. We finally provide some ideas how a richer set of observations can be included into our framework.

Acknowledgements

We would like to thank the members of the Computational Perception Laboratory for discussions and Wasinee Rungsarityotin and Gabriel Brostow for their help with writing this paper. This research was partially supported by a PhD fellowship from Microsoft Research and DARPA grant #F49620-00-1-0376.

References

[1] G. Brostow and I. Essa. Motion based decompositing of video. In *International Conference on Computer Vision*, pages 8–13, 1999.

[2] T. Darrell and A.P. Pentland. Robust estimation of a multi-layered motion representation. In *IEEE Workshop on Motion 1991*, pages 173–178, 1991.

[3] I. Haritaoglu, D. Harwood, and L.S. Davis. A fast background scene modeling and maintenance for outdoor surveillance. In *ICPR00*, pages Vol IV: 179–183, 2000.

[4] T.C.M. Lee. Segmenting images corrupted by correlated noise. *PAMI*, 20(5):481–492, May 1998.

[5] Jonathan W. Shade, Steven J. Gortler, Li-Wei He, and Richard Szeliski. Layered depth images. In Michael F. Cohen, editor, *Computer graphics: proceedings: SIGGRAPH 98 Conference proceedings, July 19–24, 1998*, Computer Graphics -proceedings- 1998, pages 231–242, New York, NY 10036, USA and Reading, MA, USA, 1998. ACM Press and Addison-Wesley.

[6] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *CVPR99*, pages II:246–252, 1999.

[7] P.H.S. Torr, R. Szeliski, and P. Anandan. An integrated bayesian approach to layer extraction from image sequences. *PAMI*, 23(3):297–303, March 2001.

[8] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *International Conference on Computer Vision*, pages 255–261, 1999.

[9] J.Y.A. Wang and E.H. Adelson. Representing moving images with layers. *Image Processing*, 3(5):625–638, September 1994.

[10] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[11] S.C. Zhu and A. Yuille. Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *PAMI*, 18(9):884–900, September 1996.