Motion Based Decompositing of Video

Gabriel J. Brostow, Irfan A. Essa

College of Computing, GVU Center,
Georgia Institute of Technology,
Atlanta, GA 30332-0280 USA
{brostow|irfan}@cc.gatech.edu
http://www.gvu.gatech.edu/perception/projects/layering/

Abstract

We present a method to decompose video sequences into layers that represent the relative depths of complex scenes. Our method combines spatial information with temporal occlusions to determine relative depths of these layers. Spatial information is obtained through edge detection and a customized contour completion algorithm. Activity in a scene is used to extract temporal occlusion events, which are in turn, used to classify objects as occluders or occludees. The path traversed by the moving objects determines the segmentation of the scene. Several examples of decompositing and compositing of video are shown. This approach can be applied in the pre-processing of sequences for compositing or tracking purposes and to determine the approximate 3D structure of a scene.

1 Introduction

We address the problem of decompositing video into planar layers. As observed from a single camera, motion within a scene can reveal the relative depths of static objects. This motion serves to segment the *3D* space, but does not have to be caused by any specific activity – human or otherwise.

From the camera's perspective, activities that occur near the lens occlude objects that are in the background. Similarly, foreground objects occlude activities taking place further from the lens. This second case is often treated as a limitation of monocular tracking systems because occlusions are not modeled explicitly. For example, a naïve system could track a person walking through a room, but might report that the person has lost their legs when they move behind a couch.

Motivated to explicitly model occlusions and relative depths of backgrounds, we borrow the concept of layers as used in the world of film compositing. The *3D* world is modeled as a stack of *2D* masks, essentially providing a *2½D* representation. While compositing processes merge existing layers of static and dynamic scenes, we seek to generate and sort the static layers according to the path followed by the active regions.

Since we use the path to generate *2½D* information from our *2D* image sequences, our layer representation will be as extensive as a given path permits. Figure 1 shows a multi-exposure image of a person walking through a scene. Figure 2 shows the extracted constituent layers.

[*]This research was funded in part by grants from the National Science Foundation #EIA-98068022 and #GER-9454185.

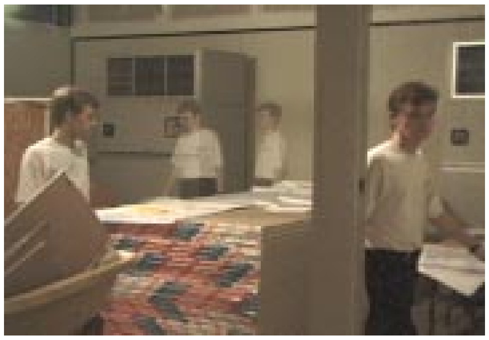


Figure 1. Time-lapse of a person moving through a complex environment (Scene newlab2).

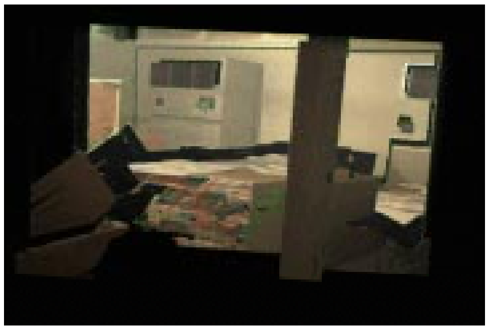


Figure 2. Stack of layers extracted by this motion (Scene newlab2).

We present several examples of automatic decomposition of video into its constituent layers. Dynamic information in these sequences consisted of a person walking through scenes of varying complexity. The resulting relative depth information is used to perform recomposition of the actual layers with synthetic ones.

Related Work: Our work benefits from several important contributions from the fields of vision and graphics. Psychological research on human visual perception shows that we utilize occlusions in determining boundaries, even in the absence of edge and brightness information [7, 6]. This serves as a primary motivation for our approach.

For the synthesis of realistic scenes, researchers in computer graphics have proposed various techniques for compositing. In most instances, compositing is performed on layers that are separated out manually or by using *chroma-keying* techniques. Smith's triangulation method stands as one of the most precise approaches to segmentation [12]. However, this method is limited when dealing with scenes containing action, since the actions would have to be exactly repeatable in front of two different backgrounds. Consequently, the special effects industry continues to use the effective, though cumbersome, approach pioneered by Vlahos to deal with blue screen (or color differencing) segmentation [15, 14]. Blue screening requires a special stage setup, which is a limiting factor for some productions, and restricts the availability of this technique to production studios.

The concept of extracting layers from video is not new to computer vision. Several motion based methods have been proposed to extract layers from moving scenes [17, 4]. In these approaches, optical flow measurement is used to classify a scene into its constituent layers. Our work differs from these in that we do not explicitly compute motion. Instead, we rely on simple changes in occlusion, caused by a moving object, to extract similar information. Figure 3 shows occlusion boundaries revealed by motion. This allows for a higher-level representation of layers as our method permits extraction of relative depth in a scene with one moving object. We also avoid the standard problems associated with optical flow [2].

Several groups have represented a static scene as a combination of layers with heightfields [1, 11, 12]. The heightfields are extracted using egomotion, where activity within the scene itself is not permitted. This is an efficient technique for compositing images to represent absolute depth information from different viewpoints. While both our approaches require some hand initialization, our method uses a single viewpoint and a moving object to extract layers with relative depth. Our method is favorable when dealing with scenes containing action, or when intra-object relationships are of interest.

Our approach is aimed at extracting static scene information by observing activity. This is a different approach than standard structure from motion, which generally aims to reconstruct a complete representation of a scene viewed by a moving camera.

Grimson *et al.* [5] present a method which uses activities to construct a model of a scene. They use information about the active object (*e.g.*, height) and its relationship to the ground-plane to compute depth. We obtain similar results without specific domain knowledge.

The techniques presented in this paper can serve as an automatic preprocessing step for tracking and synthesis of activities in complex scenes. Systems like [6] could benefit from the relative depth information we are able to obtain automatically. In the rest of this paper, we describe the details of our approach, give illustrative examples of our results, and conclude with a discussion of possible uses of our methodology.

2 Classifying Blobs

Layers consist of sets of static blobs. Our algorithm separates blobs by spatially segmenting the scene into *2D* regions, based on color features and spatial coherency. Correct assignment of these static blobs to layers comes as a result of occlusions caused by the active blobs.

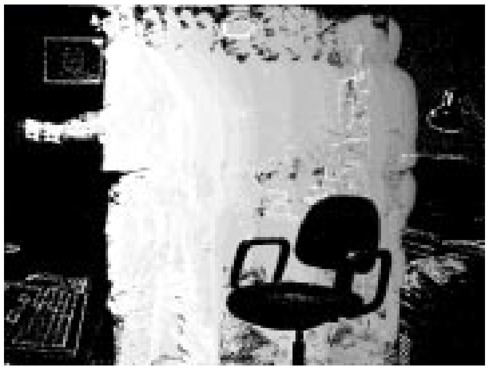
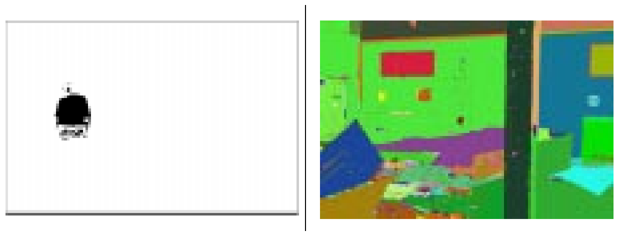


Figure 3. This shows a motion history image (MHI) of a moving person in a scene. Note how occlusion information allows us to identify the chair and its location in front of the person. For more information on motion history images see [3].

Active Blobs: All objects that move in a scene as active blobs [19, 10]. Background subtraction is performed to locate pixels whose color differs from the background image's values (see Figure 4(a)). While performing a rather clean extraction of the moving object, subtraction has an inherent limitation: moving objects occasionally have a similar color to the background pixels they pass over. Consequently, multiple blobs might be used to represent a single moving entity. Since we only require that the entity moves coherently, the multiple-blob representation is sufficient and does not affect processing.

Our strictly low-level treatment of active blobs is deliberate. We wish to make as few assumptions as possible about the potentially amorphous objects that might be moving through our scene. For the current implementation, this means that every group of active blobs is necessarily a moving object passing in front of something. A domain-specific vision system might utilize this system as a pre-processing stage, extending it with feedback and predictions regarding the activities of the active blobs. Our current objective is the accurate classification of the static blobs into layers.

Static Blobs: The same reference frame used for background subtraction is used for isolating static blobs. Figure 4(b) was initially segmented according to Laplacian spatial-edge detection. However, edge detection of just the background image reveals only some of the important edges. Many neighboring objects have no clear boundary between them if their colors are too similar. We refer to such boundaries as hidden edges. Motion between objects can reveal the hiding edges, at least temporarily, if the active blobs are of a sufficiently contrasting color. As part of our segmentation, we run through the image sequence looking for hidden edges that appear only temporarily. The scene depicted in Figure 4 appears in Figure 5 as a combination of normal spatial edges and discovered hidden ones. The moving person unhides some of these edges by revealing (over several frames) a color-contrast where there was previously none.



(a) (b)

Figure 4. (a) Active blob in newlab2 sequence which represents a partially occluded moving person. **(b)** Flood-filled static blobs from same sequence are monitored for consistency.

In many scenes this processing is still insufficient to form closed boundaries, so we performed some contour completion of the edge detected reference frames manually. Other automatic contour completion algorithms similar to [14, 9] are in our plans for future work. However, we still intend to keep manual adjustment of contour completion as a part of our implementation, since segmentation is not the focus of our work.

The information stored about each static blob includes each blob's aspect ratio and boundary zone. The boundary zone is similar to a bounding box, but is a dilated version of the blob's footprint.

Occlusion of Blobs by Other Blobs: Our approach deals with the scenarios where only the static blobs have a known shape. Nothing is assumed about the moving objects beyond their being clustered in depth. The *2D* projection of a static blob does not change, but the projection of an active blob does. Consequently, we must consider occlusion scenarios for active blobs which move in front and behind.

The simpler scenario occurs when the active blobs move in front of a static blob. For those frames of the sequence where a "significant" amount of the active blobs overlaps the static blob, the static blob is marked as occluded. Therefore, the static blob is "deeper" in the scene. The criterion used to establish if the overlap amount was significant is a pixel-count, which is scaled by the blob size and aspect ratio.

The more difficult scenario entails identifying a static blob as an occluder. Since the active blobs could be in any shape, we cannot use the same approach to determine if the active blobs are deeper. Instead, assume that moving objects have difficulty occupying the same space as a static blob's boundary zone. Eventually, the moving object should either occlude the static (the previous case), or the active object will pass behind the static.

3 Sorting out Layers

Our initial use of the concept of layers was simply an extension of the spatial relationship between blobs and pixels; Layers (L_i) are aggregates of blobs, just like blobs are spatial groupings of pixels. While blobs are the result of spatial segmentation, layers are produced through temporal segmentation.

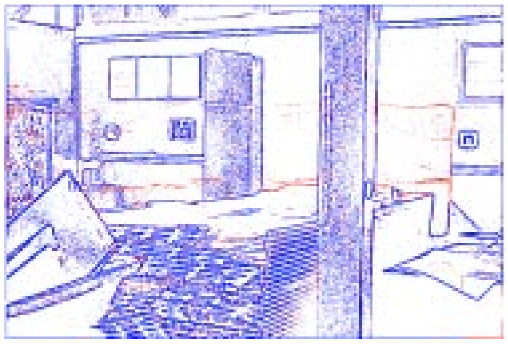


Figure 5. Hidden Edge Detection: Regular edges appear in blue, "hidden" edges appear red for sequence newlab2.

3.1 Assigning Blobs to Layers

Initially, since we have no a priori knowledge, we assume that all blobs in a scene belong to a single layer called the zero-layer (L_0). The actual physical distance between the camera and this or any other layer is unknown. However, an active blob object moving through the scene provides us with relative-depth cues that allow us to move blobs from this zero-layer to either background (L_{-i}) or foreground (L_{+i}) layers.

Occlusion of a static blob results in dropping the static blob to a background (negative) layer. This is a "push" step as depicted in the process diagram in Figure 6(c). There a toy car performs the activity and the cup is pushed. Similar processing is done in the other occlusion scenario, where the static blob is in front of the region of activity. In that scenario, the cup is moved to a foreground layer, and the active blob again remains in the zero-layer. We call this transfer of a blob to a higher layer "popping" the blob forward.

As long as the static blob is not occluded, our algorithm checks the amount of overlap between the active blobs and each static blob's boundary zone. If the overlap is sufficient, we wait until the active blobs leave the boundary zone. Figure 6(b) and 6(d) demonstrate this step of our algorithm with the cup eventually being popped. That static blob will be marked for popping to the present depth of activity only if the boundary zone overlap ceases without an occlusion of the static blob itself.

3.2 Generating Multiple Layers

So far, we have only discussed pushing and popping blobs from the L_0 to the L_{-1} and L_{+1} layers, respectively. This is sufficient for scenes that only have one foreground and one background, and where all the activity occurs between them. Here, all static blobs that are occluded end up in L_{-1}, and all blobs that acted as occluders wind up in L_{+1}. However, many scenes are more complex and feature objects moving within different fields of depth (i.e. between various layers).

We assume that activity is only occurring in one depth plane at a time, or that the active blobs are "walking" between layers. An "activity layer" represents the current depth of the moving objects relative to our *2D* and *2½D* world. At the start of a sequence when all the blobs belong

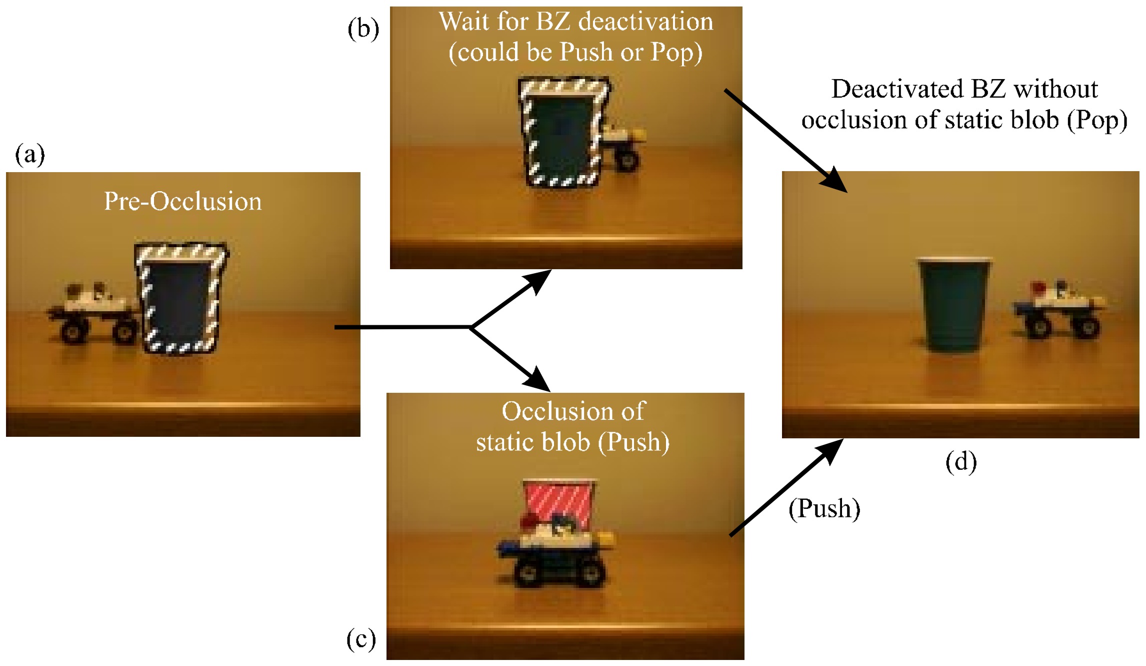

(a)
Pre-Occlusion
(b)
Wait for BZ deactivation (could be Push or Pop)
Deactivated BZ without occlusion of static blob (Pop)
(c)
Occlusion of static blob (Push)
(Push)
(d)

Figure 6. Process flow diagram with Border Zone (BZ) overlay.

to the zero-layer, the activity layer is also coplanar with the zero-layer. As long as the static blobs that get segmented as occluders or occludees are cut from the zero-layer, the active layer remains at the zero-depth. Such is the case if the moving objects move through the scene following a simple path, which does not zig-zag back and forth around the same objects.

However, many sequences consist of more complex paths. If the active blobs occlude a blob already in the foreground layer, the activity layer shifts to just in front of that foreground layer (*i.e*, from L_0 to L_{+2}). Similarly, if there is activity behind a blob in a background layer, the activity layer shifts to sit behind that background layer (from L_0 to L_{-2}). Successive pushing and popping is performed relative to the adjusted active layer's depth in the same manner as when it was in L_0 – comparing depths, inserting new layers, or re-adjusting the activity layer as necessary to maintain regularity.

3.3 Locus-Dependant Layer Correctness

Blobs in a given layer are not always closer to the camera than those in more distant layers. A path followed by a single set of active blobs will not result in a universally correct depth mapping, since there could be multiple paths through a scene, each with its own variations in layer assignment. More domain information would be necessary for a path-independent solution.

Since classification of layers is based on the *3D* path followed by the moving objects, then depth is correct at least with respect to the motion's locus. Two real objects that are equidistant from the camera will appear in two different layers if a person walks between them. Also, if the path through the scene does not monotonically increase or decrease the active blobs' distance to the camera, layer information must be stored as a function of time to accommodate paths that loop around static objects. These examples illustrate the inherent discrepancy between real-world layers and perceived layers.

However, the generated arrangement of layers is actually correct for tracking applications that perform higher level analysis of this image sequence. These time-dependent perceived layers are also useful for the tracking of other moving objects through this scene, as long as the sample moving object's size and path are representative of the other examined activities. Besides the path, the size is significant because a small object will fail to provide as much occlusion information as tracking a large object would require.

4 Compositing Using Layers

Our layer representation was conceived, in part, with compositing in mind. Instead of blue-screening, we can apply our extraction algorithm to the motion sequence after filming. Placing a blue-screen behind a static object is roughly equivalent to observing activity behind that object.

When moving among the objects in the scene, the "talent" is performing segmentation. Each resulting layer contains color information for sets of blobs that did not exhibit the ability to allow objects to pass between them. The rest of each layer is marked as transparent. We therefore have the capability to manipulate the scene at a higher level than pixels.

Since the layers are arranged in the correct order of depth, inserting a sequence with transparency information is simple. In Figure 7, we present an example in which we mapped an animation onto a synthetic layer, which did not exist in the real scene. This new virtual layer, shown featuring an animated dragon, can be inserted anywhere in the stack of true layers (*i.e*, as the most foreground, most background, or somewhere in between). The resulting re-rendering of the scene produces a composite video with a dragon. As would be appropriate to real objects at that relative depth to the camera, the dragon is hidden or revealed, albeit losing the translucent effect of the windows.

Our approach does not limit where the new layer is inserted. Changing the depth of the virtual layer over time, *i.e*, moving it higher or lower in the stack, can generate

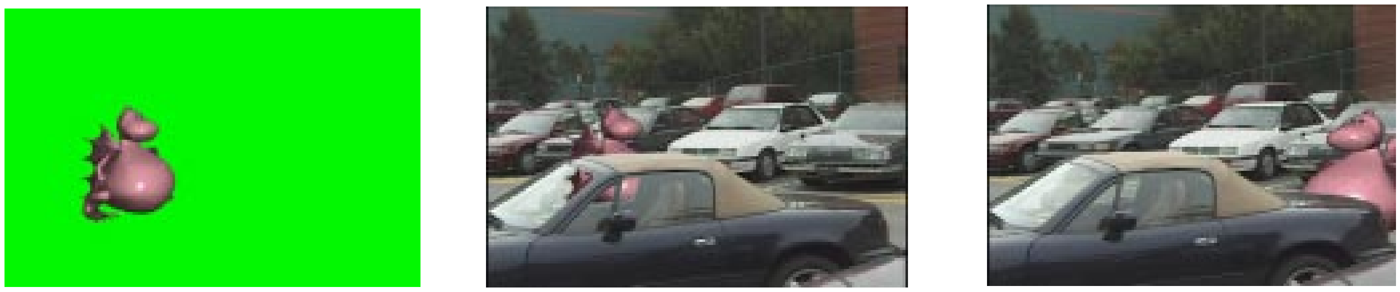

Synthetic Layer (Frame 20) Composited (Frame 20) Composited (Frame 72)

Figure 7. Synthetic layer inserted between L_0 and L_1. Frames 20 and 72 from re-composited sequence. Note that dragon is visible through the windows of the car.

interesting effects. Such manipulation of the inserted layer allows for simple motion between objects and in depth relative to the camera.

Removal of unwanted objects is done in the same manner as insertion. Here, this is achieved by pulling out a whole layer from the stack. It is important to replace the resulting transparent holes with other objects at some layer.

5 Results and Discussion

We have experimented with different extended video sequences. Each video consisted of scenes with varying complexity. In this section, we present the results of decompositing this video data. In each of the video sequences, we recorded data without any motion and then had a subject walk around in the scene.

In the following paragraphs, we discuss the results of our technique on two representative video sequences from our database of 15. Sequence “car1” is an outdoor sequence with only two layers and contains translucent objects (Figure 8). Sequence “newlab2” is a cluttered indoor sequence of a construction site filmed under low lighting conditions (Figure 9). Additional results can be viewed in color on the project web-page.

For each sequence, a background image is segmented using edge detection and contour completion. Some manual intervention is required to clean up blob boundaries. Figure 5 shows the result of our hidden-edge detection of an indoor sequence. While completing many blob boundaries correctly, some spurious edges were also generated in parts of the scene where the head and other high-contrast parts of the body persisted for five or more frames. However, the majority of these spurious edges did not form closed boundaries, and were later correctly absorbed in the interior of static blobs.

Thick edges (more than two pixels wide/tall) were more problematic because they were not explicitly assigned to specific static blobs. Such hapless edges were explicitly ignored during layer-generation, as were static blobs that were too small (area less than 10 pixels) to monitor accurately. Thick edges, small blobs, and blobs that were never affected by the activity in the scene all remain in the unclassified layer, L_0. These appear in our sample composite sequences as artifacts, which are visible if the animated dragon moves under L_0. The blobs that were too far from the active blobs’ path are not errors as they are the result of a lack of occlusion information. They will continue to be ignored until activity in an extended sequence exposes them as occluders or occludees.

The algorithm is designed to break the depth scene up into its smallest constituent parts. By definition, blobs are texture-based components of the scene, and do not necessarily have a one-to-one correspondence with objects. As a result, the same object can be split into two separate layers if the observed activities occurred with enough time-separation. These split layers are adjacent to each other in the stack. Since we do not prevent someone from mistakenly placing a new layer between the split layers during post-production, overlooking these split layers could cause consistency problems. The problems could be avoided by detecting and merging the split layers manually.

An unforeseen limitation of this work is the significance of reflections and shadows. The sequence car2 (not pictured), which is similar to car1, erroneously assigned a car from the foreground to the background layer. Upon closer analysis, we discovered that the person walking behind that car was both reflecting in its metallic paint-finish, and was casting a shadow on its hood. That limitation of our approach that will require special handling.

6 Conclusions and Future Work

A method for decompositing video into its constituent planar layers is presented. These layers are sorted based on their relative depths from the camera. The depth of these layers is determined by detecting the occlusion events caused by a moving object (in our case a person). As the moving object traverses the scene, the depth of the spatially segmented blobs in the scene is adjusted. Spatial information is obtained through edge detection and a contour completion algorithm. In situations where parts of the scene match the color of the moving objects, some manual augmentation is required. The extracted layers provide an effective representation for compositing and tracking applications.

Our current implementation produces a stack of layers that appear as billboards in front of a static camera. This representation provides an intuitive way of inserting and deleting layers for special effects applications. We show decomposition of several image sequences as well as the compositing of a synthetic layer into a real scene.

We envision several future extensions. We are investigating several approaches that will deal with the limitations caused by shadows and reflections. Camera-motion compensation and multi-camera processing will be added to allow our technique to aid in *2½D* or *3D* reconstruction when the cameras move or zoom. Also, activity recognition can be added to allow for monitoring of multiple moving objects at a time.

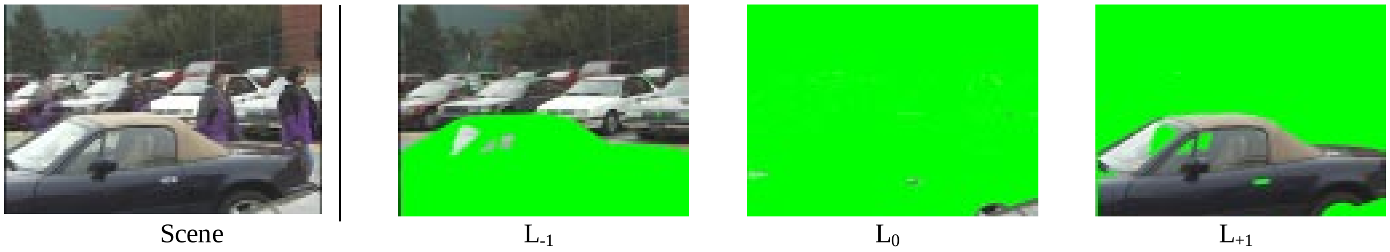


Figure 8. Multi-exposure from original car1 sequence and 3 extracted layers.

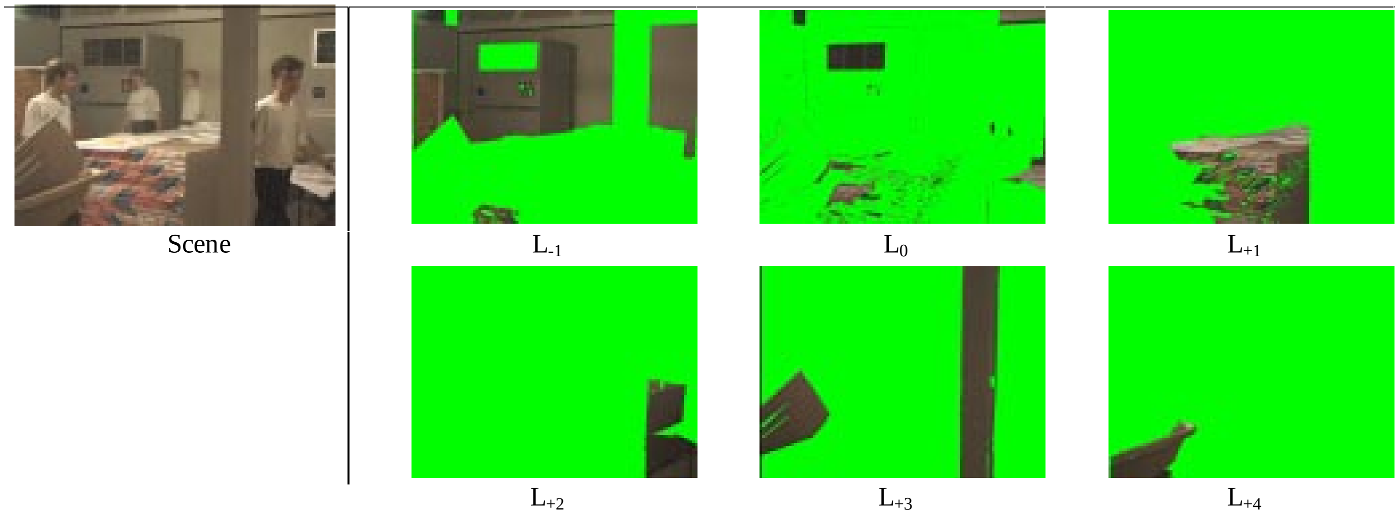


Figure 9. Multi-exposure from original newlab2 sequence and 6 extracted layers.

Acknowledgements

The authors wish to thank Arno Schödl, Drew Steedly, and Tony Haro for code fragments and numerous discussions which helped bring this work to fruition.

References

[1] S. Baker, R. Szeliski, P. Anandan. A Layered Approach to Stereo Reconstruction. In *CVPR98*, pages 434 – 441, 1998.

[2] J.L. Barron, D.J. Fleet, and S.S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43 – 77, February 1994.

[3] A. F. Bobick and J. W. Davis. An apearance-based representation of action. In *Proceedings of International Conference on Pattern Recognition 1996*, August 1996.

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

[5] W.E.L. Grimson, L. Lee, R. Romano, and C. Stauffer. Using adaptive tracking to classify and monitor activities in a site. In *CVPR98*, pages 22 – 31, 1998.

[6] Youichi Horry, Ken ichi Anjyo, and Kiyoshi Arai. Tour into the picture: Using a spidery mesh interface to make animation from a single image. In Turner Whitted, editor, *SIGGRAPH 97 ConferenceProceedings*, Annual Conference Series, pages 225–232. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[7] G. Kanizsa. Organization in vision: Essays on gestalt perception. In *Praeger*, 1979.

[8] K. Nakayama. Biological image motion processing. *Vision Research*, 25:625–660, 1985.

[9] T. Pavlidis. Curve fitting as a pattern recognition problem. In *ICPR82*, pages 853–859, 1982.

[10] S. Sclaroff and J. Isidoro. Active blobs. In *ICCV98*, pages 1146–1153, 1998.

[11] 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.

[12] H. Shum, R. Szeliski, S. Baker, M. Han, and P. Anandan. Interactive 3d modeling from multiple images using scene regularities. In *Proceedings of European Workshop SMILE98*, pages 236 – 252, 1998.

[13] Alvy Ray Smith. Blue screen matting. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 259 – 268. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[14] S. Tehrani, T.E. Weymouth, and B.G. Schunck. Interpolating cubic spline contours by minimizing second derivative discontinuity. In *ICCV90*, pages 713 – 716, 1990.

[15] URL. http://www.ultimatte.com.

[16] P. Vlahos. Comprehensive electronic compositing system. U.S. Patent no. 4,100,569, 1977.

[17] B. Wandell. *Foundations of Vision*. Sinauer Associates, 1995.

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

[19] 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.