IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 30, no. 7, p. 1171–1185, July 2008.

Motion Segmentation and Depth Ordering Using an Occlusion Detector

Doron Feldman, and Daphna Weinshall, *Member, IEEE Computer society*

***Abstract*— We present a novel method for motion segmentation and depth ordering from a video sequence in general motion. We first compute motion segmentation based on differential properties of the spatio-temporal domain, and scale-space integration. Given a motion boundary, we describe two algorithms to determine depth ordering from two- and three-frame sequences. A remarkable characteristic of our method is its ability compute depth ordering from only two frames. The segmentation and depth ordering algorithms are shown to give good results on 6 real sequences taken in general motion. We use synthetic data to show robustness to high levels of noise and illumination changes; we also include cases where no intensity edge exists at the location of the motion boundary, or when no parametric motion model can describe the data. Finally, we describe psychophysical experiments showing that people, like our algorithm, can compute depth ordering from only two frames, even when the boundary between the layers is not visible in a single frame.**

***Index Terms*— Computer vision, Video analysis, Motion, Depth cues, Segmentation**

1. Introduction

THE goal in motion-based segmentation is to partition images in a video sequence into segments of coherent motion. There are two main approaches: some assume a global parametric motion model and segment the image according to the parameters of the model (e.g., [12], [23], [24], [34]), whereas others assume piecewise smooth motion and identify the boundaries along motion discontinuities (e.g., [3], [13], [21], [33]). The second approach is potentially more general, and it lies at the base of our proposed method here.

Motion discontinuities can be identified by clustering a previously computed motion field. The problem is that such discontinuities are found at exactly those locations where the computation of the motion field is least reliable: since all optical flow algorithms rely on the analysis of a region around a point (even if only to compute first-order derivatives), the optical flow must be continuous within the region to support reliable computation. This chicken-and-egg problem, which can be addressed in different ways (e.g., [24], [34]), makes motion segmentation particularly challenging. On the other hand, the successful computation of motion discontinuities can be useful for a number of applications, including motion computation (by highlighting those areas where the computation should be considered unreliable) and object segmentation from multiple cues. Here we propose a motion segmentation method that does not require a reliable optical flow to begin with.

Having segmented the image, we next want to determine the occlusion order of objects in the image, as the first step in 3D scene understanding and object recognition. In principle, any depth-retrieval algorithm (e.g., [13]) would also provide depth ordering. However, full 3D reconstruction is usually only practical in static scenes, and it relies on accurate geometric calibration which remains a hard task. In this work we present a method to compute depth ordering from occlusion cues without explicit scene reconstruction. The most important characteristic of our method is its ability compute depth ordering from only two frames.

The problem of depth ordering is similar to figure/ground segregation, an issue which has been studied extensively in the context of Gestalt psychology. Many possible spatial cues may contribute to figure perception from a single image, including *convexity* [25], *junctions* [27], and *familiar configurations* [26]). However, depth ordering from a single image may be subjective and prone to ambiguities, whereas motion gives a very powerful and usually unambiguous cue.

Given an image sequence, the accretion and deletion of texture elements [11], as well as the *common fate* of texture and edge [5], [36], have long been recognized as cues for depth ordering. There are several methods for depth ordering from three frames or more, e.g., by tracking disappearing texture elements [20], optical flow filling [24], detecting T-junctions in space-time [1], [22], matching the motion of surface and boundary [4], [6], [31] and localization of errors in flow computation w.r.t. monocular segmentation [2].

However, as we claim in Section 5, when given only two frames, it is impossible to determine depth ordering from motion alone, without additional assumptions or prior knowledge. This is because the motion of pixels that become occluded cannot be determined, and thus they may belong to either side of the motion edge, leading to more than one valid order assignment. One solution would be to assume that the occluded pixels belong to the layer that is more similar in appearance; i.e., determine depth ordering by matching the motion of color and motion edges [32]. However, color edges are often unreliable as edges between layers, since the figure and ground may have similar colors.

1.1. Motion Segmentation

Our work is based on the extraction of motion boundaries, which are defined *locally* as boundaries between different motions (since many real video sequences do not obey any global motion model). Several methods rely on color or texture edges [8], [13], [30], which can be combined with alpha matting to produce precise results [35]. In this work we restrict ourselves to solutions which do not rely on such spatial cues, which are not always present at motion boundaries. This is further motivated by humans' ability to segment objects from motion alone (e.g., in random dot kinematograms), and by the need to avoid over-segmentation of objects whose appearance includes varying color and textures. Finally, we only consider local properties of the

The authors are with the School of Computer Science and Engineering, the Hebrew University of Jerusalem, 91904 Jerusalem, Israel. E-mail: {doronf,daphna}@cs.huji.ac.il.

temporal profile of motion, in order to be able to deal with pairs of frames or stereo pairs (but see, for example, [29], [35]).

In our approach, originally reported in [7], we start by considering the video sequence as a spatio-temporal intensity function, where the goal is to extract information from this spatio-temporal structure. Video sequences have highly regular temporal structure, with regions of coherent motion forming continuous tube-like structures. These structures break where there is occlusion, creating spatio-temporal corner-like features. Using a differential operator that detects such features, we develop an algorithm that extracts motion boundaries.

Specifically, our algorithm is based on the occlusion detector described in Section 2.1. This operator is used to extract a motion boundary at any given scale, as described in Section 2.2. Since different scales may be appropriate for different parts of the image, a cross-scale optimal boundary is computed, based on the response of the detector. At the end, a closed contour is built along the most salient boundary fragments to provide the final segmentation. The algorithm was evaluated on three challenging real sequences, as described in Section 3. We included a number of synthetic examples which are particularly difficult for some commonly used algorithms, in order to demonstrate the robustness of our method. Some Results from other algorithms, whose implementation was made available by the authors, are provided for comparison. Finally, in Section 4 we analyze the behavior and mathematical properties of the algorithm.

1.2. Depth Ordering

A few recent papers explicitly model occlusion based on matching, or lack thereof [13], [35], which can be used to infer depth ordering. In this work we introduce a novel low-level cue that indicates depth order.

Our computational approach to the problem of ordinal depth from two frames utilizes the principle of common fate of texture and boundary, though without attempting to extract the boundary explicitly. The spatio-temporal partial derivatives in each frame are affected by both the motion of the layers (i.e., their texture), and the motion of the motion boundary. When using our occlusion detector, which relies on these derivatives, a bias towards the occluded side appears. The bias depends on the density gap between the two layers (this bias disappears when the layers have the same local density). Moreover, when measuring this bias in scale space, it can be seen to increase as the scale is increased.

From this observation we derive an algorithm in Section 5.1, which computes the ordinal depth of two layers based on the trend of the bias in scale-space. With some minor modifications, we show in Section 5.2 that the same algorithm can be applied to three-frame sequences, without relying on local differences of density between the layers. The algorithms are shown to perform well on real sequences. The performance of the algorithms is compared to the performance of human subjects on two- and three-frame sequences of random-dot textures of varying density and to an ideal observer model in Section 6.

2. Segmentation Algorithm

The motion segmentation algorithm we present is based on a differential operator defined in Section 2.1 that is applied to the video sequence and responds at motion boundaries. While this operator is shown to detect motion boundaries in many cases, it is often unable to detect boundaries where certain degeneracies exist locally. This is solved by a cross-scale scheme presented in Section 2.2. Finally, closed contours are extracted using a saliency measure and a simple heuristic to overcome small gaps, presented in Section 2.3. See also Appendix II for some implementation issues.

2.1. Occlusion Detector

Regarding the video sequence as a spatio-temporal intensity function, let $I(x,y,t)$ denote the intensity at pixel (x,y) in frame t. We refer to the average of the second moment matrix over a neighborhood ω around a pixel as the *Gradient Structure Tensor*

$$\mathbf{G}(x,y,t) \equiv \sum_{\omega} \nabla I \ (\nabla I)^T = \sum_{\omega} \begin{bmatrix} I_x^2 & I_x I_y & I_x I_t \\ I_x I_y & I_y^2 & I_y I_t \\ I_x I_t & I_y I_t & I_t^2 \end{bmatrix} \quad (1)$$

This matrix has been invoked before in the analysis of local structure properties. In [14], eigenvalues of $\mathbf{G}$ were used for detecting spatio-temporal interest points. In [18] it was suggested that the eigenvalues of $\mathbf{G}$ can indicate spatio-temporal properties of the video sequence and can be used for motion segmentation. The idea behind this is reminiscent of the Harris corner detector [9], as it detects 3D "corners" and "edges" in the spatio-temporal domain. Here we take a closer look and develop this idea into a motion segmentation algorithm.

Specifically, if the optical flow in ω is (v_x, v_y) and the brightness constancy assumption [10] holds, then

$$\mathbf{G} \cdot (v_x, v_y, 1)^T = 0 \quad (2)$$

Hence, 0 is an eigenvalue of $\mathbf{G}$. Since $\mathbf{G}$ is positive-semidefinite, we can use the smallest eigenvalue of $\mathbf{G}$ as a measure of deviation from the assumptions above, which leads to the following definition:

Definition 1: Let $\lambda(x,y,t)$ denote the smallest eigenvalue of the *Gradient Structure Tensor* $\mathbf{G}(x,y,t)$. The operator λ is the *occlusion detector*.[1]

We do not normalize λ with respect to the other eigenvalues of $\mathbf{G}$ (as in [18]), since it may amplify noise. In order to provide rotational symmetry and avoid aliasing due to the summation over the neighborhood ω, we define ω to denote a Gaussian window, and the operation $\sum_{\omega}$ in (1) stands for the convolution with a Gaussian. Since we do not assume temporal coherence of motion, the Gaussian window is restricted to the spatial domain.

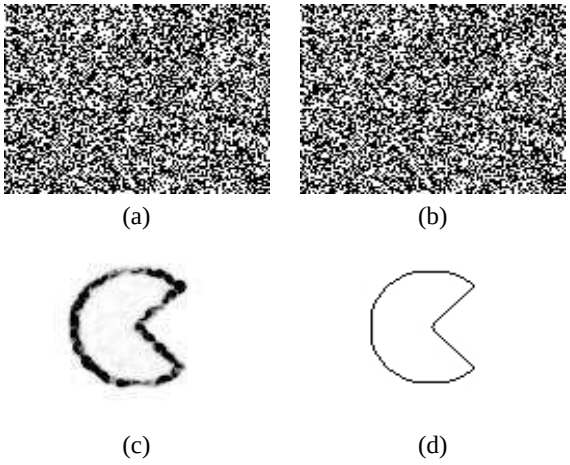
Figure 1 demonstrates the detector results on a simple synthetic example. In this example there are no intensity or texture cues to indicate the boundaries of the moving object, and it can only be detected using motion cues. The value of λ, shown in Fig. 1c, is low in regions of smooth motion, and high values of λ describe the boundary of the moving object accurately.

The values of ∇I, and hence of λ, are invariant to translation transformations on I. Additionally, for any rotation matrix $\mathbf{R}$,

$$|\lambda\mathbf{I} - \mathbf{G}| = |\mathbf{R}(\lambda\mathbf{I} - \mathbf{G})\mathbf{R}^T| = \left| \lambda\mathbf{I} - \sum_{\omega} (\mathbf{R}\nabla I)(\mathbf{R}\nabla I)^T \right|$$

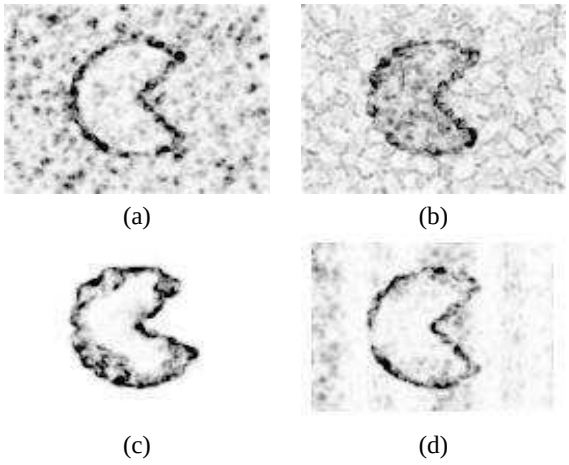
($\mathbf{I}$ is the identity matrix) and therefore the values of λ are also invariant to the rotation of I. The issue of scale invariance is discussed in Appendix I.

[1] Note that the values of λ at each pixel can be evaluated directly using Cardano's formula.



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

Fig. 1. Random dots example. A shape is moving sideways, where both the shape and the background are covered by a random pattern of black and white dots. It is impossible to identify the moving object from each of the two frames (a) and (b) (a stereo pair) alone. The occlusion detector (c) (higher values of λ are darker) shows the outline of the object very clearly. Compare to the ground truth (d).



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

Fig. 2. False λ response. The same example as in Fig. 1: (a) with 20% white noise; (b) with illumination change of 5%; (c) with the object rotating by 20°; (d) with both object and background patterns deformed smoothly.

Velocity-adapted detector: Although rotational invariance is desirable in the spatial domain, non-spatial rotations in the spatio-temporal domain have no physical meaning. It is preferable to have invariance to spatially-fixed shear transformations, which correspond to 2D relative translational motion between the camera and the scene. As suggested in [15] by the reference to *Galilean diagonalization*, one can use the velocity-adapted matrix $\tilde{\mathbf{G}}$ given by

$$\tilde{\mathbf{G}} = \begin{bmatrix} G_{11} & G_{12} & 0 \\ G_{21} & G_{22} & 0 \\ 0 & 0 & \lambda_T \end{bmatrix} \quad \text{where} \quad \lambda_T = \frac{\det(\mathbf{G})}{\det(\mathbf{G}^*)} \qquad (3)$$

(G_{ij} denote the entries of $\mathbf{G}$, and $\mathbf{G}^*$ denotes the 2×2 upper-left submatrix of $\mathbf{G}$ containing only spatial information).

Definition 2: The operator λ_T is the *velocity-adapted occlusion detector*.

To justify this definition, observe that $\tilde{\mathbf{G}}$ is also invariant to translation and spatial rotation. The entry λ_T is an eigenvalue of $\tilde{\mathbf{G}}$, and it has been suggested that it encodes the temporal variation, being the "residue" unexplained by pure-spatial information.

In practice, λ_T gives results similar to λ, though it has certain advantages, as discussed in Section 4. Throughout this paper we use λ to denote either operator, unless stated otherwise.

Detector effectiveness: High values of λ indicate significant deviation from (2), which is often due to the existence of a motion boundary. Other sources of large deviations include changes in illumination (violation of the brightness constancy assumption), or when the motion varies spatially (motion is not constant in ω). However, often these events lead to smaller λ values as compared with motion boundaries (see Fig. 2), in which case the boundary response can be distinguished from a false response (e.g., by thresholding).

Low values of λ do not necessarily indicate that the motion in ω is uniform. The rank of $\mathbf{G}$ is affected by spatial structure as well as temporal structure, so λ may be low even at motion boundaries, when certain spatial degeneracies exist. Specifically, this occurs when there is local ambiguity, i.e., when the existence of a motion boundary cannot be determined locally. This includes areas where the occluding object and its background are of the same color, areas where the background is uniform in color, and areas where the background texture is uniform in the direction of the motion (Fig. 3). In the first case the rank of $\mathbf{G}$ is 0, and in the other cases the rank of $\mathbf{G}$ may be 1 or 2, depending on the appearance of the occluding object (recall that the λ detector is high when the rank of $\mathbf{G}$ is 3). In these cases, the background may be interpreted as part of the moving object, since no features in the background appear to vanish due to occlusion.

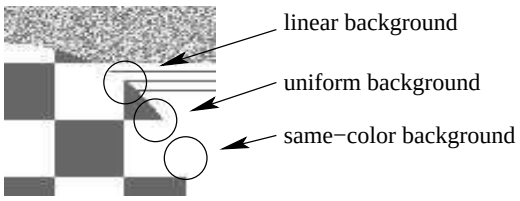


Fig. 3. Areas where the λ detector is likely to give low values despite the existence of a local motion boundary.

2.2. Extraction of Motion Boundaries and Scale Space Structure

The response of λ to occlusion occurs only where some background features become occluded. Clearly boundary location cannot always be inferred on the basis of local information alone. However, while there may be no cues to indicate the location of the boundary at a fine scale, there may be enough information at a coarser scale (i.e., in a larger neighborhood) and λ may respond. Thus we incorporate a multi-scale element in our algorithm, in order to detect motion boundaries that are not detectable at fine scales.

Defining scale: In order to define the notion of scale in our algorithm, note that the evaluation of λ involves Gaussian convolutions in two different stages – during the estimation of the partial derivatives, and when taking the average over the neighborhood ω. In both cases, larger Gaussians lead to coarser structures, and we refer to the size of the Gaussian as the *scale*. In this work we only consider the spatial scale. As we show in Appendix I, these two scales are related, and we define a unified scale dimension, and a scaling-invariant operator $\lambda^{(s)}$ at any scale $s > 0$, using scale-normalization.

(a) (b)

(c) (d)

Fig. 4. Checkerboard example: (a) A frame from the sequence; (b) and (c) show the response of λ at fine ($s_{xy} = 1$) and coarse ($s_{xy} = 10$) scales respectively. At the fine scale, λ only responds at intensity edges (which appear as discrete "bursts"), whereas the entire contour is visible at the coarse scale, though with considerable distortion. (d) shows the final contour selected by integrating over scales.

The notion of scale has been studied extensively for features such as edges and blobs. As with these features, different structures can be found at different scales. The response of λ to noise, which can occur in finer scales, is suppressed in coarser scales. On the other hand, localization is poor at coarse scales and motion boundaries may break and merge.

Figure 4 illustrates this idea – at fine scale (Fig. 4b), λ responds only at discrete locations, because the background consists of regions with constant color, and the occlusion can only be detected where there are color variations in the background. In the coarser scale (Fig. 4c), the neighborhood of every boundary point contains gradients in several directions and the boundary is detected continuously.

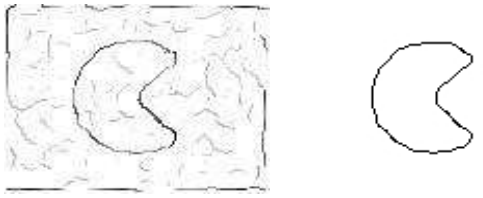
Image features, such as edges, typically shift and become distorted at coarse scales. The scale space structure of motion boundary edges (and in particular our occlusion detector) has its own particular biases in coarse scales. As discussed in Section 4, motion boundaries at coarse scales are shifted towards the occluded side, i.e., the occluding objects becomes "thicker". In addition, it can be shown that the bias is stronger when there is a large intensity difference between the object and the background, and it increases with scale.

Estimating derivatives in the temporal domain is prone to aliasing. See Appendix II for implementation details, including elimination of aliasing and estimation from only two frames.

Boundary extraction in scale space: Since λ is computed by taking the average over a neighborhood, its response is diffuse. We want to extract a ridge curve where λ is strongest. This can be defined locally as points where λ is maximal in the direction of the maximal principal curvature, which can be expressed as

$$\begin{cases} \lambda_{xy}(\lambda_x^2 - \lambda_y^2) - \lambda_x\lambda_y(\lambda_{xx} - \lambda_{yy}) = 0 \\ (\lambda_{xx}+\lambda_{yy}) \cdot \left((\lambda_{xx}-\lambda_{yy})(\lambda_x^2-\lambda_y^2)+4\lambda_x\lambda_y\lambda_{xy}\right) < 0 \\ \lambda_x^2\lambda_{yy} - 2\lambda_x\lambda_y\lambda_{xy} + \lambda_y^2\lambda_{xx} < 0 \end{cases} \quad (4)$$

Thus, at every scale s, the values of λ and its derivatives are computed, and the ridge can be extracted. For reasons of numerical stability, the derivatives of $\lambda^{(s)}$ are computed with the same Gaussian smoothing s used for computing $\lambda^{(s)}$, at each scale.



(a) (b)

Fig. 5. Saliency measure. (a) All boundaries extracted from the random dots example with illumination changes (Fig. 2b); intensity codes λ response. (b) The most salient closed contour.

Different boundaries are extracted at different scales, as fine-scale boundaries may often split because of the absence of local information, and coarse-scale boundaries may disappear or merge. Since these may occur at different parts of the image at different scales, we need to construct a scale-adapted boundary, by selecting different scales for different localities (as in [16]). Considering the multi-scale boundary surface as the union of all ridges in $\lambda^{(s)}$ for $s \in (0, \infty)$, we want to find a cross-scale boundary where $\lambda^{(s)}$ is maximal. This can be expressed as

$$\begin{cases} \lambda_s = 0 \\ \lambda_{ss} < 0 \end{cases} \quad (5)$$

using the scale derivatives of λ.

Combining (4) and (5) defines the final *cross-scale motion boundary*. It is a curve in the three-dimensional space $X-Y-S$, defined by the intersection of the two surfaces defined respectively by these 2 sets of equations.

2.3. Boundary Completion

As stated above, λ also has some false responses which lead to the selection of false boundary fragments. It is therefore necessary to define a saliency criterion, which is used to select the most interesting boundaries. Since we regard λ as a measure of local boundary strength, for each connected set of boundary points we define the *saliency measure* to be the sum of the value of λ along the boundary, as in [16]. This measure may be sensitive to fragmentation of the boundary, so in our implementation we tolerate small gaps.

Finally, segmentation is achieved by searching for closed contours with high saliency and small gaps using a simple heuristic method. Since the extracted boundaries are usually almost complete, this heuristic gives good results (see Fig. 5).

The algorithm starts by finding a closed region with high saliency. The detected edges are thickened so as to bridge over small gaps (5 pixels are typically sufficient), thus segmenting the image into regions. For each such region, the saliency of its bordering edges is summed and the most salient region is selected. Finally, considering only those edges that border the selected region, we employ a simple heuristic method to connect the motion boundary fragments into a continuous boundary with maximal saliency and minimal gaps.

3. Experimental Results

In our experiments we applied our algorithm to a few sets of real and synthetic image pairs. The running time of the MATLAB implementation for 256×192 images is approximately 70 seconds, and is roughly linear in the number of pixels.

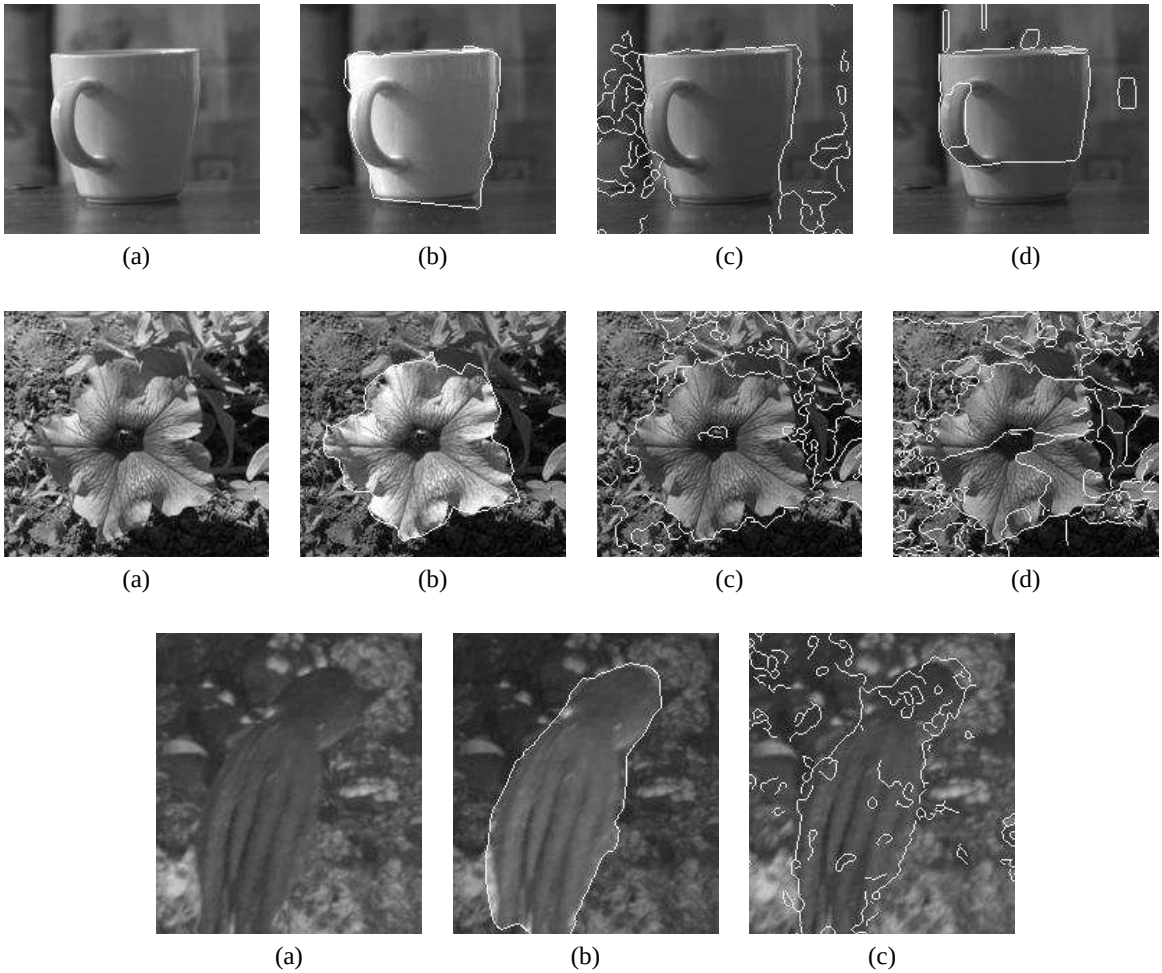


Fig. 6. Results on real sequences. The cup (top) and flower (middle) examples are stereo pairs, and the octopus sequence (bottom) is a dynamic scene. (a) One of the frames. (b) The most salient edge detected by our algorithm (with the area of the segment highlighted). (c) Canny edges in the optical flow. (d) Edges from a MRF-based segmentation algorithm (Kolmogorov and Zabih [13]).

We compared our algorithm with the most prominent motion segmentation approaches, wherever code was available. To begin with, we establish the baseline result by segmenting the optical flow. Such a segmentation lies at the heart of some more elaborate segmentation methods, such as [24]. We used a robust and reliable implementation of the Lucas-Kanade algorithm [17], and segmented it using a variety of edge operators, including Canny and various anisotropic diffusion methods and clustering methods (e.g., [34]), presenting the best results for each example.

One influential motion segmentation approach is based on Markov Random Fields [13] (and is therefore related to the more traditional regularization based approaches [19]). Code for two variants of this approach is available on the web by the respective authors [13], [30], and we could therefore use their code to establish credible comparisons. We note, however, that in both cases the publicly available code can only work with rectified images. Therefore, in order to obtain fair comparisons, we compared our results with the results of these algorithms only with rectified image pairs, when possible.

The cup and flower examples in Figure 6 demonstrate our algorithm's performance on a stereo pair. The most salient motion boundary is shown in Fig. 6b superimposed on the first input image. Fig. 6c illustrates the baseline result - the edges of the optical flow. Fig. 6d illustrates the best MRF-based segmentation using graph cuts [30].

The octopus example in Figure 6 shows our algorithm's performance on a video sequence with a dynamic scene, featuring non-rigid motion and illumination changes. The octopus and the reef below have similar color and texture, and thus spatial coherence is unreliable (note in particular the triangle-shaped projection near the octopus' head, which is in fact a background feature).

In Fig. 7, a large amount of noise was added to the synthetic checkerboard sequence, causing numerous optical flow estimation errors. The magnitude of the flow estimation error is often greater than the true flow (Fig. 7b), particularly around the centers of the squares, making segmentation based directly on the optical flow impossible. Results of our algorithm and MRF-based method are also shown.

The main weakness of many MRF-based methods is their reliance on spatial coherence, which leads to failure when no spatial edge coincides with the motion edge. This is demonstrated on the random dots example in Fig. 8a,b where such methods have no spatial support and therefore fail. Fig. 8c,d demonstrates our algorithm's advantage when no global motion model can be assumed. In this example, the texture of both the moving object and the background undergo smooth non-linear deformation. The results of applying [34] show that when motion varies smoothly within an object, global model methods fail.

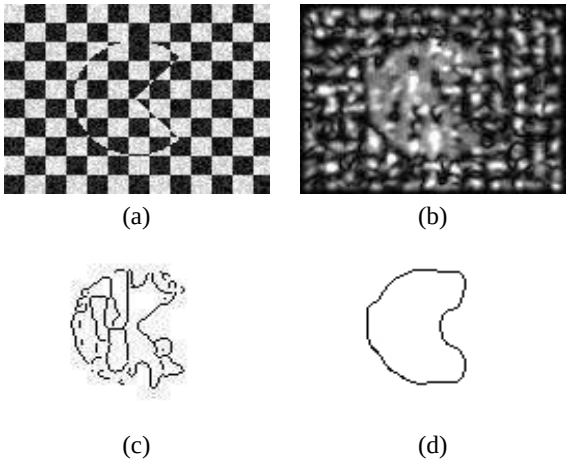


Fig. 7. Checkerboard example with 25% white noise. (a) One of the frames; (b) Lucas-Kanade optical flow magnitude; (c) MRF-based segmentation; (d) The most salient contour found by our algorithm.

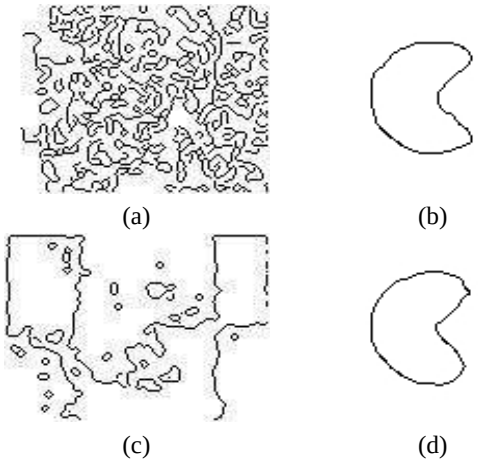


Fig. 8. Random dots example (see Fig. 1). With 20% white noise: (a) MRF-based segmentation; (b) The most salient contour found by our algorithm. With smooth non-linear deformation: (c) Segmentation assuming affine motion using an implementation of [34]; (d) The most salient contour found by our algorithm.

Figure 9 demonstrates how our algorithm works with very slow motion. As long as there are features in the background that become occluded, our algorithm can detect the motion boundary even at sub-pixel motion. Figure 9a shows results for a sequence where the foreground object moves by $1/2$ pixel. All MRF-based algorithms we applied failed to detect the foreground object altogether. Although the velocity in Fig. 9a is 8 times slower than that in Fig. 9b, the values of λ in both cases are similar.

Figure 10 shows results on the synthetic Yosemite sequence, which consists of a terrain with no occlusions that has non-rigid motion (in 2D), and a cloud pattern with illumination changes. The detector response is very weak in the terrain region, and the motion edge between the terrain and the sky is correctly detected.

4. Analysis

In order to analyze the performance of the proposed technique, we consider a video of two moving layers l^1, l^2, where w.l.o.g. l^2 partially occludes l^1. A frame in the video sequence can be written as

$$I = l^1 \cdot (1 - m) + l^2 \cdot m \tag{6}$$

Fig. 9. Results on a random dots example with small motion of $1/2$ pixel per frame (a), and with larger motion of 4 pixels per frame (b).

Fig. 10. Results on the synthetic Yosemite sequence.

where m is the *matting map*. We assume w.l.o.g. that the occlusion edge is perpendicular to the X axis and that at frame $t = 0$ it is at $x = 0$. We further assume that the occlusion edge is a Gaussian-smoothed line, so m is of the form $m_{s_0}(x) = \int_{-\infty}^{x} g_{s_0}(u)du$ (we denote the Gaussian function with variance s as g_s).

If the motions of l^1 and l^2 are (v_x^1, v_y^1) and (v_x^2, v_y^2) respectively, then the video volume is given by

$$\begin{aligned} I(x,y,t) \quad = \quad & l^1(x - v_x^1 t, y - v_y^1 t) \cdot (1 - m(x - v_x^2 t)) + \\ & l^2(x - v_x^2 t, y - v_y^2 t) \cdot m(x - v_x^2 t) \end{aligned} \tag{7}$$

Note that the motion of m is the same as the motion of l^2, since it is the occluding layer.

Denoting the video volume of each layer as $I^k(x, y, t) = l^k(x - v_x^k t, y - v_y^k t)$, the gradient of the video volume is given by

$$\nabla I \quad = \quad (1 - m) \cdot \nabla I^1 + m \cdot \nabla I^2 + (I^2 - I^1) \cdot g_{s_0} \cdot \mathbf{n} \tag{8}$$

where $\mathbf{n} = (1, 0, -v_x^2)^T$. Note that $\mathbf{n}$ is perpendicular in space-time to the occlusion edge $(0, 1, 0)^T$ and to the motion vector $\mathbf{v}^2 = (v_x^2, v_y^2, 1)^T$; i.e., $\mathbf{n}$ is the normal to the plane in the video space formed by the motion of the occlusion edge.

Therefore, ∇I is composed of the matting of ∇I^1, ∇I^2, and a component that depends on $I^2 - I^1$. Note that ∇I^1 is perpendicular to $\mathbf{v}^1$, whereas both ∇I^2 and $\mathbf{n}$ are perpendicular to $\mathbf{v}^2$. This means that ∇I is composed of two components that are related to the occluding layer and only one that is related to the occluded layer.

For scale space analysis we use the approximation

$$g * (f \cdot m) \approx (g * f) \cdot (g * m) \tag{9}$$

where g is a Gaussian function and m is an integral of a Gaussian as defined above. Eq. (9) is an equality when f is constant, and it provides a good approximation when f does not change rapidly near $x = 0$ (in each layer separately).

Applying (9), the gradient estimated at scale s, denoted by $\nabla I^{(s)} = \nabla(g_s * I)$, is

$$\begin{aligned} \nabla I^{(s)} \quad \approx \quad & (1 - m_{s_0+s}) \cdot \nabla I^{1(s)} + m_{s_0+s} \cdot \nabla I^{2(s)} + \\ & (I^{2(s)} - I^{1(s)}) \cdot g_{s_0+s} \cdot \mathbf{n} \end{aligned} \tag{10}$$

4.1. Velocity-Adapted Occlusion Detector λ_T

We assume the 2D gradients in each layer are distributed isotropically, in the sense that the mean gradient is 0. Furthermore, we assume that they are uncorrelated. Thus, using (8) and (9), we can write the gradient structure tensor defined in (1) as

$$\begin{aligned}\mathbf{G}^{(s)} &\approx g_{s_\omega} * \Big((1-m_{s_0+s})^2 \nabla I^1 (\nabla I^1)^T + \\ &\quad m^2_{s_0+s} \nabla I^2 (\nabla I^2)^T + I^2 - I^1)^2 \cdot g^2_{s_0+s} \cdot \mathbf{n}\mathbf{n}^T\Big) \\ &\approx h_1 \cdot \mathbf{M}^1 + h_2 \cdot \mathbf{M}^2 + h_3 \cdot \mathbf{n}\mathbf{n}^T \qquad (11)\end{aligned}$$

where

$$\mathbf{M}^k \equiv \begin{bmatrix} 1 & 0 & -v_x^k \\ 0 & 1 & -v_y^k \\ -v_x^k & -v_y^k & (v_x^k)^2 + (v_y^k)^2 \end{bmatrix} \qquad (12)$$

and

$$\begin{aligned} h_1 &= c_1 \cdot (1 - m_{s_0+s+s_\omega})^2 \\ h_2 &= c_2 \cdot m^2_{s_0+s+s_\omega} \\ h_3 &= c \cdot g_{s_\omega + (s_0+s)/2} \end{aligned} \qquad (13)$$

The coefficients $c = \left\langle (l^2 - l^1)^2 \right\rangle / \sqrt{4\pi(s+s_0)}$ and $c_k = \left\langle \|\nabla l^k\|^2 \right\rangle / 2$ describe the distribution of intensities in the layers.

Then, the velocity-adapted occlusion detector from (3) can be shown to be

$$\lambda_T = \frac{(v_x^1 - v_x^2)^2}{1/h_1 + 1/(h_2+h_3)} + \frac{(v_y^1 - v_y^2)^2}{1/h_1 + 1/h_2} \qquad (14)$$

Maximum: In the general case, the expression above is hard to analyze. Simulations show that λ_T typically has a single local maximum. Although it may have two local maxima, this only happens when $c_2 > 9 \cdot c_1$ and $c > 180 \cdot c_1$ for $s \geq 1$, and the second local maximum is usually very subtle. Therefore, for all practical purposes, it can be assumed that λ_T has a single local maximum.

Bias due to texture: In the limit $c \to 0$ (i.e., both layers have similar intensities), λ_T becomes

$$\lambda_T = \frac{(v_x^1 - v_x^2)^2 + (v_y^1 - v_y^2)^2}{1/c_1(1-m)^2 + 1/c_2 m^2} \qquad (15)$$

Differentiating for m yields that λ_T is maximal at x_{max} such that

$$m(x_{max}) = \frac{\sqrt[3]{c_1}}{\sqrt[3]{c_1} + \sqrt[3]{c_2}} \qquad (16)$$

and thus $x_{max} > 0 \iff c_1 > c_2$, which means that the location of the detected edge is biased towards the layer with lower intensity variance. The magnitude of the bias vanishes when $c_1 = c_2$, and it is proportional to $\sqrt{s + s_0 + s_\omega}$, therefore it vanishes at fine scales.

Bias due to occlusion: In the case where $c > 0$ and $c_1 = c_2$ (i.e., both layers have the same intensity variance), the detected edge location is biased towards the occluded layer. To see this, we substitute $x = 0$ in the derivative of λ_T

$$\begin{aligned}\frac{d\lambda_T}{dx}(x=0) &= \frac{-(v_x^1 - v_x^2)^2}{\sqrt{\pi(s_0+s+s_\omega)} \cdot (\sqrt{s_\omega + s_0 + s}/c + \sqrt{2}/c_1)} \\ &< 0 \qquad (17)\end{aligned}$$

Since λ_T is always positive, has a single local maximum x_{max}, and vanishes at $x \to \pm\infty$, it follows that $\frac{d\lambda_T}{dx} > 0$ when $x < x_{max}$ and $\frac{d\lambda_T}{dx} < 0$ when $x > x_{max}$. From (17) it follows that $x_{max} < 0$, which means that the detected edge location is biased towards the occluded layer.

4.2. Occlusion Detector λ

Behavior analysis of the smallest eigenvalue λ is harder. Thus we make the further assumption that $l^1 = l^2$ along the edge. Then we can omit the last term in (11) and get

$$\mathbf{G} = c_1(1-m)^2 \mathbf{M}^1 + c_2 m^2 \mathbf{M}^2 \qquad (18)$$

Calculating the eigenvalue of (18), the following can be shown:

- The smallest eigenvalue of $\mathbf{G}$ is given by

$$\lambda = \frac{1}{2}\left(a - \sqrt{a^2 - 4b}\right) \qquad (19)$$

  where

$$\begin{aligned} a &= (1-m)^2 c_1 \|\mathbf{v}^1\|^2 + m^2 c_2 \|\mathbf{v}^2\|^2 \\ b &= (1-m)^2 m^2 c_1 c_2 \|\mathbf{v}^1 - \mathbf{v}^2\|^2 \end{aligned}$$

- λ has a single local maximum.
- If $c_1\|\mathbf{v}^1\|^2 = c_2\|\mathbf{v}^2\|^2$, then λ is maximal at $x = 0$ – where the edge is located.
- If $c_1\|\mathbf{v}^1\|^2 > c_2\|\mathbf{v}^2\|^2$, then λ is maximal at some $x > 0$, and vice-versa; in other words, the detected edge location is biased towards the layer with lower intensity variance and smaller absolute motion.

The biasing effect towards the occluded layer is not evident due to the particular assumption we have made, although it was observed in our experiments. Note that λ is affected by absolute velocity, unlike the velocity-adapted operator λ_T.

4.3. Discussion

The analysis we have presented, albeit approximate and limited to an idealized model, explains properties of the occlusion detectors that are observed with real sequences in a much wider scope. Assuming that the occlusion edge is linear approximates the local behavior of smooth edges (or any edge in coarse scale), and empirical evidence suggests that the behavior of the detector at corners is also similar. Finally, assuming that the edge is aligned with the Y axis clearly does not limit generality, due to rotation invariance.

In general, the distribution of intensity gradients also does not significantly affect the properties discussed above, although strong features may affect localization in their vicinity. Even though we have analyzed the biases due to texture and due to occlusion separately, clearly they may occur together.

One important aspect of motion segmentation that was not addressed in this analysis is the mutual effect of different edges, possibly from different objects, on each other. Edges shift at coarser scales and ultimately merge, which limits the applicability of this approach at coarse scales.

5. Depth Ordering

We now present two algorithms for determining ordinal depth based on the occlusion detector defined in Section 2.1, using either two frames or three frames.

When only two frames are available, it is impossible to infer the order of depth from motion alone, without additional assumptions or prior knowledge. Consider a pair of images of a video sequence

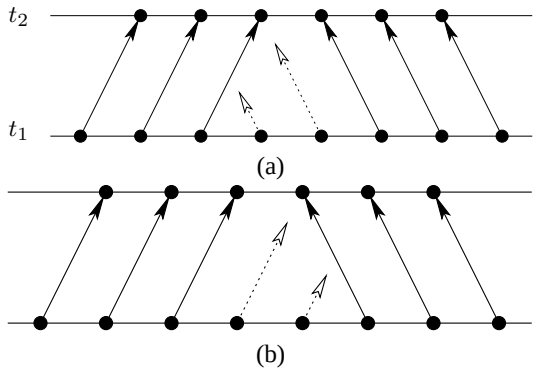


Fig. 11. Two-frame occlusion problem. Two of the pixels in frame t_1 do not correspond to any pixel in t_2 due to occlusion, and they may belong either to the right (a) or the left (b) layer.

(or a stereo pair) that contain the motion of two layers where one partially occludes the other. As illustrated in Fig. 11, pixels that appear in one frame and become occluded in the other may belong to either of the layers. Whichever layer they belong to is the occluded layer, and since their interframe correspondence cannot be determined, both interpretations are equally valid. Our two-frame algorithm, described in Section 5.1, is based on the assumption that there is a (possibly small) difference of intensity between the layers on the average.

The situation when more than two frames are available is considerably different. Although there may be two interpretations to a two-frame sequence, additional frames can be used to rule out false interpretations. With a slight modification, our algorithm can be applied to three frames even when the two layers have the same intensity on the average, and achieve better localization, see Section 5.2.

5.1. Two-Frame Algorithm

Given the scenario described above and generalizing (8), the space-time gradient of I is given by

$$\nabla I \quad = \quad \nabla I^1 \cdot (1-m) + \nabla I^2 \cdot m + (I^2 - I^1)\nabla m \quad (20)$$

Observe that the expression above is a sum of three vectors – two of them proportional to the gradients of the two layers, and a third component that stems from the edge between the layers. Since the edge and the occluding layer have the same motion (or *common fate*), the gradient of I is more affected by the motion of the occluding layer than that of the occluded layer in areas of transition between layers. This asymmetry is manifested in a *bias* towards the occluded layer in the location of the detected motion boundary, as derived from (14).

We first note that this bias typically grows with scale. This is because the components representing the gradients of each layer are smoothed across the motion boundary into the other layer, and the component that is due to the difference between the layers is smoothed in both directions. Therefore, the effect of the motion of the occluding layer expands farther into the occluded layer as I is further smoothed.

More specifically, consider the spatial scaling of a video I by σ, namely

$$J(x,y,t) = I(x/\sigma, y/\sigma, t) \quad (21)$$

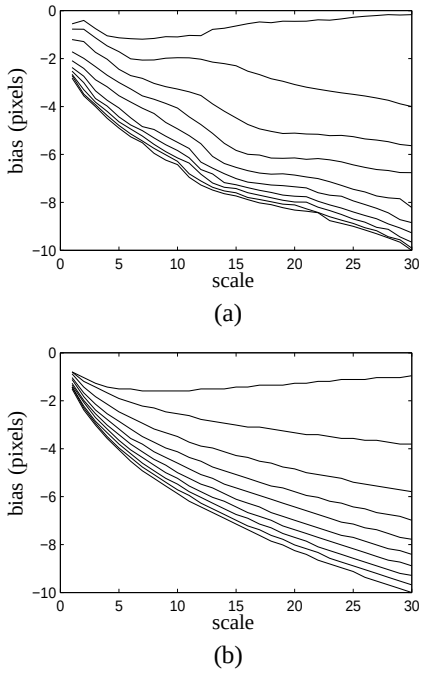


Fig. 12. (a) The bias of the location of maximal $\lambda^{(s)}$ as a function of s (scale) on a synthetic random-dot pair. Each curve represents a different value of $\left\langle (I^2 - I^1)^2 \right\rangle$ ranging from 0 (top) to 0.2 (bottom). (b) The bias as predicted based on (14).

(a) (b)

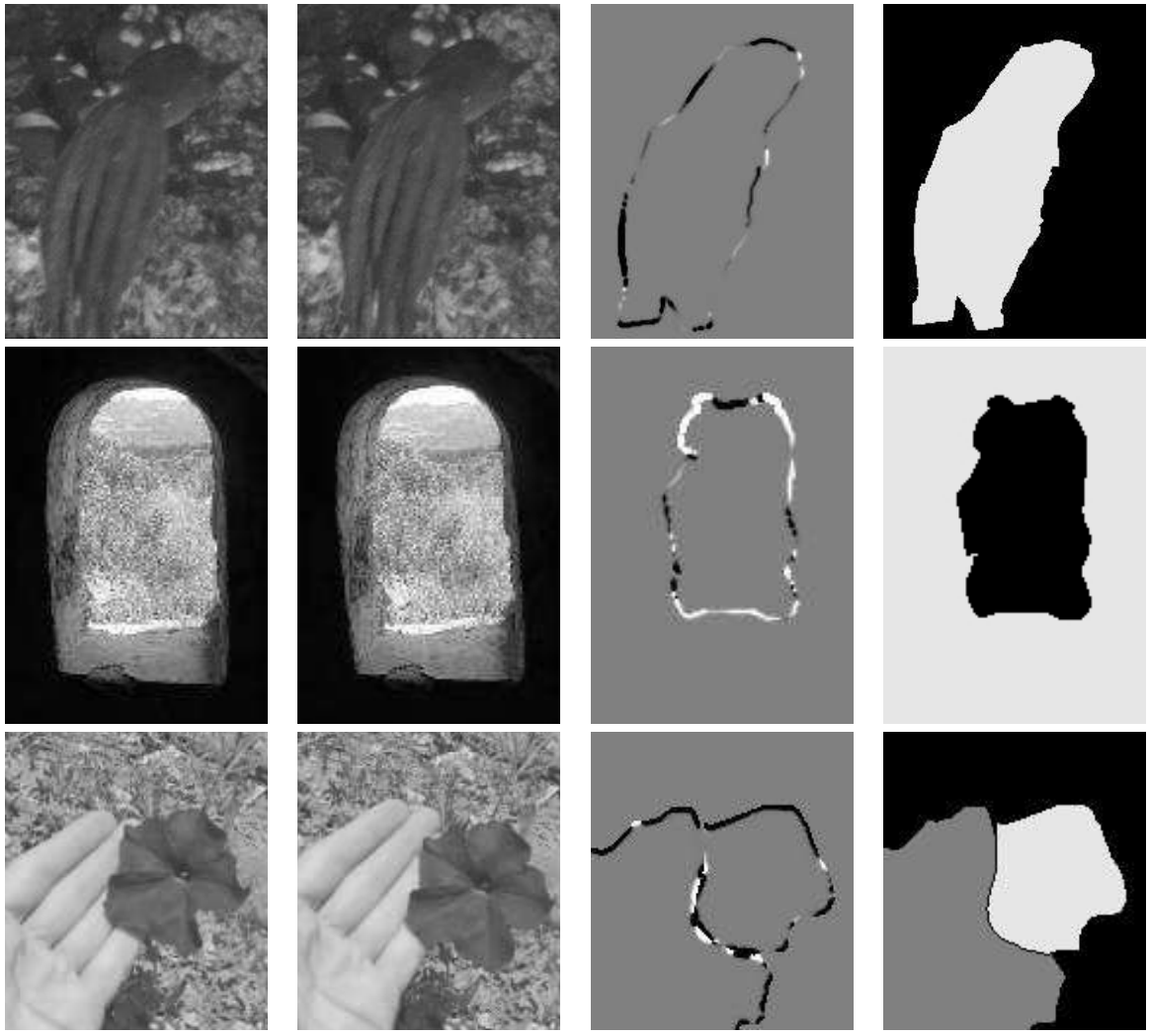
Fig. 13. Random dots example with 20% density difference between foreground and background: the edge of $\hat{m}$ is superimposed on the response of λ at scale $s = 5$ with an occluding (a) and occluded (b) segment

Due to scaling invariance (31),

$$\lambda_J^{(\sigma^2 s)}(x,y,t) = \lambda_I^{(s)}(x/\sigma, y/\sigma, t) \quad (22)$$

Thus, if at scale s_1 the maximum of $\lambda_I^{(s_1)}$ is obtained at some $x < 0$, then at scale s_2 the maximum of $\lambda_J^{(s_2)}$ would be obtained at $\sqrt{s_2/s_1} \cdot x$ when J is a scaling of I by s_2/s_1. If the values of c_1, c_2, c (defined in (13)) do not vary between the scales s_1 and s_2, then $\lambda_J^{(s_2)} \approx \lambda_I^{(s_2)}$ and the maximal λ for I at scale s_2 would also be at $\sqrt{s_2/s_1} \cdot x$. This means that the bias in the location of maximal $\lambda^{(s)}$ is proportional to $\sqrt{s}$, which means that not only is the location biased towards the occluded side, but this bias also grows with scale. This property of λ is demonstrated in Fig. 12 on a synthetic example of random dots. In real sequences, the assumption that the intensity distribution is similar in different scales is usually not satisfied. Nevertheless, the effect described above is still observed qualitatively, and can be used to determine depth ordering.

This observation can be used to design a depth-ordering algorithm. Consider the ridge e of $\lambda^{(s_1)}$ and the response of $\lambda^{(s_2)}$, for scales $s_1 < s_2$. The direction of $\nabla\lambda^{(s_2)} = (\lambda_x^{(s_2)}, \lambda_y^{(s_2)})$ is



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

Fig. 14. Results on real sequences of three dynamic scenes: (a),(b) The two frames. (c) Response of d (from Eq. (23)) coded as dark=negative, light=positive. (d) Final layers detected by the algorithm with relative depth coded as white=near, grey=middle, and black=far.

towards the ridge of $\lambda^{(s_2)}$, and therefore for each pixel along e, the vector $\lambda^{(s_2)}$ indicates the direction of the bias of $\lambda^{(s_2)}$ w.r.t. $\lambda^{(s_1)}$.

Our algorithm starts by segmenting the two-frame sequence using the segmentation algorithm described in Section 2, to yield an estimate of the matting map $\hat{m}(x,y)$ (as defined in (6)). Since the bias grows with scale, the ridges in $\lambda^{(s)}$ at higher scales should be biased with respect to the edge of $\hat{m}$. Therefore, at points along the edge of $\hat{m}$, the direction of $\nabla\lambda$ should be towards the outside if the segment is the occluder, and towards the inside if it is occluded (see Fig. 13). Defining

$$d = \nabla\lambda \cdot \nabla\hat{m} \tag{23}$$

we expect that $d < 0$ if the segment is the occluder, and $d > 0$ if it is occluded. Thus, summing the value of d along a contour of the segment can determine which side of the contour is the occluder.

Since the bias effect grows with scale, it is preferable not to use small scales. On the other hand, higher scales distort the image data and other nearby image features may interfere with the value of d. Therefore, we sum the value of d in several intermediate scales:

$$D = \sum_{s=s_1}^{s_2} \sum_{x \in \partial\hat{m}} \nabla\lambda^{(s)} \cdot \nabla\hat{m} \tag{24}$$

The response of d (from Eq. (23)) on boundary pixels in real sequences is shown in Fig. 14c. In the bottom row, points on the edge between the flower and the hand have positive values with respect to the hand and negative values with respect to the flower. Relative depth is shown in Fig. 14d. The octopus in the top row and flower in the bottom row are correctly detected as the occluders, while the hand is detected as occluding the background and as occluded by the flower. The scene viewed through the window of the old ruin in the middle row is correctly detected as occluded. Note that the internal frame of this window is (correctly) not detected, since there is no depth discontinuity in this area.

5.2. Three-Frame Algorithm

Recall that high values of λ occur in areas where there is no smooth motion, i.e., at motion boundaries. At points with no correspondence (due to occlusion), the partial derivatives would

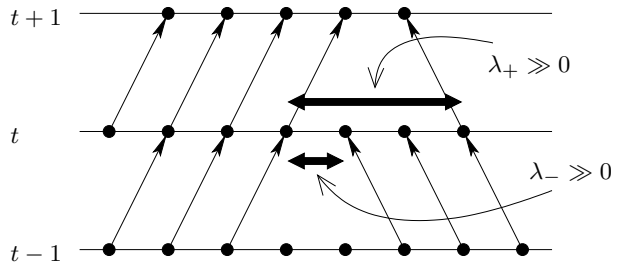


Fig. 15. Three frames with pixel correspondence; pixels that have correspondences between t and $t-1$ and have no correspondence between t and $t+1$ are located to the right of motion boundary pixels, indicating that the right side is the occluded side.

have random values (with the exceptions that were discussed in Section 2.1), leading to a high λ value, even though these points are not strictly boundary points. These areas are adjacent to the true motion boundary and the λ response would appear as a thick boundary region. Based on two frames alone, it is impossible to determine which side of the thick boundary is the true edge, which is equivalent to determining which side the occluded pixels belong to.

When three frames are available, we denote the response of λ on frames $(t, t-1)$ as λ_-, and frames $(t, t+1)$ as λ_+; t is the reference frame in both cases. We define

$$\lambda_{min} \equiv \min\{\lambda_-, \lambda_+\} \qquad \text{and} \qquad \lambda_{max} \equiv \max\{\lambda_-, \lambda_+\} \quad (25)$$

Points on the true motion boundary are detected by both λ_- and λ_+, thus $\lambda_{min} \gg 0$ at these points. Points that are not occluded in any of the frames are not detected by λ, thus $\lambda_{min} \approx 0$. There exist points that are occluded in $t-1$ and not in $t+1$ and vice versa, and in these points $\lambda_{min} \approx 0$ and $\lambda_{max} \gg 0$.

Therefore, the true motion boundary can be detected as the area where $\lambda_{min} \gg 0$. The regions where $\lambda_{min} \approx 0$ and $\lambda_{max} \gg 0$ belong to the occluded layer, and the relation between these regions and the boundary yields depth ordering, as illustrated in Fig. 15.

This approach is closely related to [24], which also uses information from the preceding frame to fill in missing information with respect to the succeeding frame, and vice versa. Here this is done implicitly based on the response of the occlusion detector, only for the purpose of depth ordering and without first extracting accurate optical flow.

This principle can be implemented by slightly modifying the two-frame algorithm as follows:

- Use λ_{min} for the segmentation to obtain $\hat{m}$.
- Use λ_{max} in (23) to obtain the bias direction d.

Using λ_{min} for the segmentation gives better localization of the segment's edge, since it responds only to the true edge. Since λ_{max} responds also to occluded regions, its profile is biased towards the occluded side (as is the bias due to the intensity gap), and thus $d < 0$ if the segment is the occluder, and $d > 0$ if it is occluded.

Unlike the bias due to intensity difference, the bias that is due to occluded pixels is not affected by scale. Note that no intensity difference was assumed, so this bias can be detected even when there is no intensity difference between the layers. On the other hand, when there is an intensity difference, both effects contribute to the bias, boosting the correct assignment.

(a) (b)

Fig. 16. Results of a real three-frame sequence (octopus example from Fig. 14): (a) Edges based on λ_{min} (black) compared to the response of λ_{max} (gray) – the response is stronger outside the edge, indicating that the segment is the occluding layer; (b) Edges based on λ_{min} (black) compared to edges based on λ_+ (i.e., from two frames), showing that three frames give better localization.

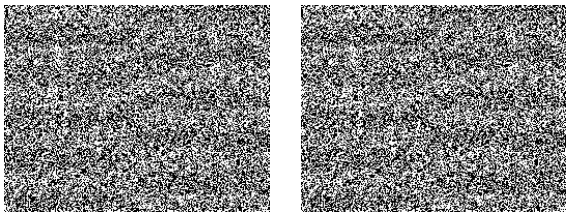


Fig. 17. Two frames used in our experiment with density varying between 45% and 55%. The sequences used in the experiment are available on the web at `http://www.cs.huji.ac.il/~daphna/demos.html#motion`.

An additional advantage of the three-frame algorithm is better localization of the segment boundary, as occluded pixels are distinguished from boundary pixels.

Figure 16a shows the edges based on λ_{min} and λ_{max} from three frames of the octopus sequence. The λ_{max} edge is outside the λ_{min} edge, indicating that the segment is the occluder. The λ_{min}-based edge gives better localization of the motion boundary (compared to the two-frame result), as shown in Fig. 16b.

6. Psychophysical Experiments

The algorithms we have presented determine the depth order from two or three frames based on motion alone. They perform well even when monocular segmentation is impossible. Below we show that human observers can also perform these tasks, with comparable success.

In Section 6.1 we describe the 2-alternative forced choice experiment, in which we presented subjects with random-dot sequences of two moving layers. In Section 6.2 and 6.3 we describe the results of experiments with two- and three-frame sequences, respectively.

6.1. Methods

In our experiments we presented subjects with sequences in which two layers with random-dot textures, one partially occluding the other, are moving horizontally in opposite directions. The boundary between the layers is the middle vertical line, and the density of the dots varies across each layer along the motion boundary. Figure 17 shows an example of such a sequence. Each side was the occluder in half of the sequences, in random order (counter-balanced).

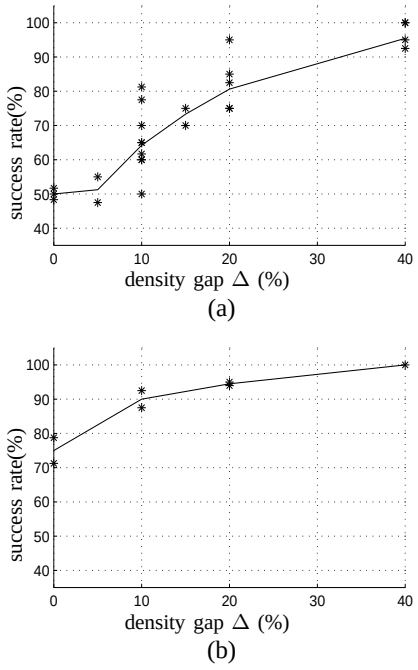


Fig. 18. Results of experiments on human subjects: (a) Two-frame sequences. (b) Three-frame sequences.

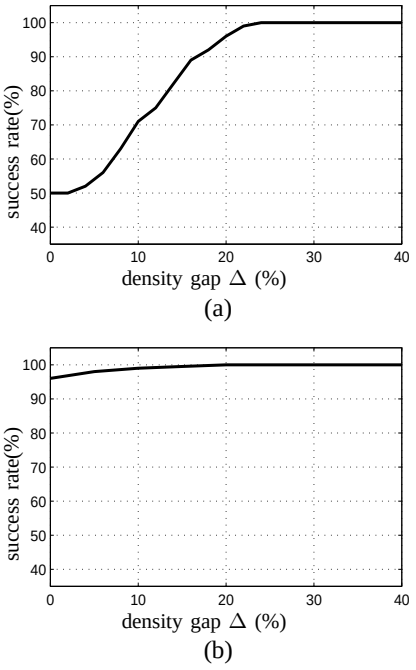


Fig. 19. Performance of our algorithm on the experiment sequences: (a) Two-frame sequences. (b) Three-frame sequences.

In each sequence, the density was characterized by some density gap Δ, so that the density varied between $(1-\Delta)/2$ and $(1+\Delta)/2$ across each layer. Participants were instructed to click on the side (left or right) where they thought the occluder was in each sequence. The experiments were conducted in sessions of 20 presentations, with 3-6 sessions per participant for each different value of density gap.

6.2. Two-Frame Sequences

Seven volunteers participated in this experiment. In each presentation, the two frames were displayed alternately at a rate of 3 frames/second. The density gap between the two frames was $0\%, 5\%, 10\%, 15\%, 20\%, 40\%$.

For a density gap of 40%, subjects selected correctly in nearly 100% of the sequences. For a density gap of 0%, i.e., the density was uniform across the whole frame, subjects selected correctly in 50% of the sequences, i.e., no better than chance. This is consistent with the fact that both interpretations are equally valid in this case. The results are summarized in Fig. 18a.

For comparison, we applied our two-frame algorithm to the same sequences. For density gaps of more than 20%, the success rate was nearly 100%. As expected, when density was uniform, the success rate was 50% (in such sequences both interpretations are equally valid). The performance of the algorithm is summarized in Fig. 19a.

6.3. Three-Frame Sequences

Two volunteers participated in this experiment. In each presentation, a sequence was played back and forth at a rate of 10 frames/second. The density gap between the two frames was $0\%, 10\%, 20\%, 40\%$. Results for three frames were much better than those for two frames, as expected. In particular, for a density gap of 0% (uniform density), subjects selected correctly in 75% of the sequences, in contrast to the two-frame experiment in which subjects performed no better than chance. The results are summarized in Fig. 18b.

Our three-frame algorithm, applied to the same sequences, gave the correct answer in nearly 100% of the sequences, and even with uniform density, its success rate was 96% (see Fig.19b).

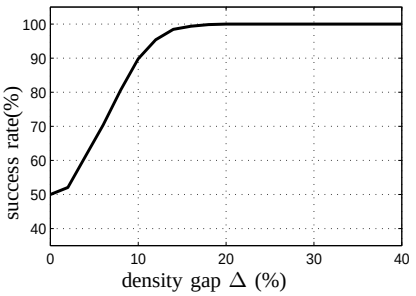


Fig. 20. Performance of an ideal observer on two-frame experiment sequences.

6.4. Two-Frame Sequences: Ideal Observer Analysis

In order to evaluate the results of the two-frame experiments and algorithm, we consider an ideal observer that "knows" the form of the distributions generating the sequences, but does not know which side is the occluder. Let H_1, H_2 denote the two possible choices: "left-front" and "right-front". For a given two-frame sequence I, the probability that it was generated as H_i is

$$\Pr(H_i|I) = \frac{\Pr(I|H_i)\cdot\Pr(H_i)}{\sum_k \Pr(I|H_k)\cdot\Pr(H_k)} \tag{26}$$

where

$$\Pr(I|H_i) = \prod_{x,y,t} \Pr(I(x,y,t)|H_i) \tag{27}$$

$\Pr(I(x,y,t)|H_i)$ and $\Pr(H_i)$ are known to the ideal observer. Thus, for any given I, the ideal observer can compute (26) for $i = 1, 2$, and then choose the most probable hypothesis. By sampling sequences, we estimated the probability of correct choice at 97.7% for $\Delta = 10\%$ and 100% for $\Delta = 20\%$. This provides a theoretical upper bound on the performance of an observer in this task.

A less informed observer, that does not know the exact form of the distribution used to generate the data, may consider all possible videos in which the density of dots in each layer remains constant within a small region. Such an observer can compare the density in neighborhoods of occluded pixels with nearby neighborhoods within either layer. For a neighborhood width of 16 pixels, such an *ad hoc* scheme chose correctly in 88% of the sequences for $\Delta = 10\%$, and 99.7% for $\Delta = 20\%$ (see Fig. 20).

7. Summary and Discussion

The occlusion detector we have presented is useful for extracting motion boundaries. Since we do not make any assumptions regarding the color or texture properties of objects, or about the geometric properties of the motion, our algorithm works well on natural video sequences where such assumptions are often violated.

The algorithm relies mainly on background features which disappear and reappear as a result of occlusion. These features may be sparse and still indicate the location of motion boundaries, as the algorithm processes the data in multiple scales. As opposed to algorithms that rely on motion estimation, our algorithm usually does not require any texture on the occluding object.

Since occlusion is the main cue used by our algorithm, it works well when velocity differences between moving objects are small, since features will still disappear due to occlusion. Algorithms that rely on motion differences typically find it hard to distinguish between different objects in such cases.

We described a second algorithm, extending the occlusion detector to compute the depth ordering between the layers across the motion boundary. The algorithm was shown to give good results on real sequences with different occlusion settings. With only two frames, the algorithm relies on some (possibly small) difference in texture between the moving layers. Without this assumption, we face the well known inherent motion ambiguity, which states that depth ordering cannot be computed from two-frames and motion alone.

Can humans use a similar heuristic to get around this inherent ambiguity? We asked humans to rank the relative depth of two moving layers in two or three frames. In our experiments there was a difference in texture between the moving layers, but the difference was set to be local and small, so that it could not be detected in a single frame as a distinct boundary between the two layers. Nevertheless, when presented to human subjects in motion, this difference was sufficient for the detection of relative depth. We showed that our algorithm can also utilize this small difference to detect relative depth, giving qualitatively similar results (cf. Fig. 18 and Fig. 19).

Appendix I
Scale Normalization

One problem with multi-scale analysis is that derivatives decrease with scale. Indeed, if $0 \le I \le 1$, then

$$|I_x|, |I_y| \le \frac{1}{\sqrt{2\pi s_{xy}}} \tag{28}$$

when smoothing with a spatial Gaussian of variance s_{xy}. This well-known problem can be handled by scale normalization, as proposed in [16]. Scale normalization is done by defining the *scale-normalized* partial derivatives

$$\begin{array}{rcl} I_x^{(s_{xy})} & = & \sqrt{s_{xy}} \cdot \frac{\partial}{\partial x}(g_{s_{xy}} * I) \\ I_y^{(s_{xy})} & = & \sqrt{s_{xy}} \cdot \frac{\partial}{\partial y}(g_{s_{xy}} * I) \end{array} \tag{29}$$

where $g_{s_{xy}}*$ stands for convolution with a Gaussian with variance s_{xy}. Thus $I_x^{(s_{xy})}$ and $I_y^{(s_{xy})}$ are used in the evaluation of λ instead of I_x and I_y. Note that scale normalization does not violate the assumptions leading to the definition of λ in Section 2.1.

One important property of scale normalization is that λ becomes invariant to spatial scaling of I. This means that λ gives comparable values for a video sequence in different resolutions.

To see this, let us scale I by σ, and define

$$J(x,y,t) = I(x/\sigma, y/\sigma, t) \tag{30}$$

Substituting (30) into (29) yields

$$\nabla J^{(\sigma^2 s_{xy})}(\sigma x, \sigma y, t) \quad = \quad \nabla I^{(s_{xy})}(x,y,t) \tag{31}$$

Let s_ω denote the variance of the Gaussian window ω, and let $\mathbf{G}^{(s_{xy}, s_\omega)}[I]$ denote the second moment matrix defined in (1), with the scale of differentiation s_{xy} and scale of averaging s_ω. From (31) it follows that

$$\left(\mathbf{G}^{(s_{xy}, s_\omega)}[I]\right)(x,y,t) = \left(\mathbf{G}^{(\sigma^2 s_{xy}, \sigma^2 s_\omega)}[J]\right)(\sigma x, \sigma y, t) \tag{32}$$

That is to say, if J is a scaling by σ of I, then the value of λ at (x,y,t) in I at scales s_{xy}, s_ω will be the same as at the corresponding point in J at scales $\sigma^2 s_{xy}, \sigma^2 s_\omega$.

For our purpose of computing a good *occlusion detector*, it follows from (32) that as long as our computation scans all scales in scale space, the result does not depend on the image resolution. Note that in order for λ to be scale-invariant, it follows from (32) that s_ω must be proportional to s_{xy}, as in [14]. In our implementation we use $s \equiv s_{xy} = s_\omega$, which defines a single scale s. We denote the λ evaluated at scale s as $\lambda^{(s)}$.

Appendix II
Implementation Issues

II.1. Temporal Aliasing

Since real video data is discrete, the partial derivatives in the definition of λ must be estimated. This is done by convolving I with the partial derivatives of a 3-dimensional Gaussian. Rotational invariance implies that the spatial variance in the X and Y directions should be the same, and the kernel is therefore an ellipsoidal Gaussian with spatial variance s_{xy} and temporal variance s_t. Due to the distortion introduced by the convolution, it is desirable that these values be small.

Estimating the temporal partial derivative from video presents a severe aliasing problem. Since video frames represent data accumulated during short and sparse exposure periods, and since a

feature may move several pixels between two consecutive frames, data is aliased in the temporal domain significantly more than in the spatial domain. We overcome this problem by taking advantage of the spatio-temporal structure of video, as described next.

Suppose that the velocity in a certain region is $v = (v_x, v_y)$, and therefore

$$I(x, y, t) = I(x - v_x t, y - v_y t, 0) \tag{33}$$

The temporal derivative in $t = 0$ is given by

$$I_t = -v_x I_x - v_y I_y \tag{34}$$

In discrete video, I_t can be estimated by convolution in the T direction, which, due to (33), is the same as convolution in the v direction of a subsample of $I(x, y, 0)$ at intervals of size $|v|$. In order to avoid aliasing due to undersampling while estimating I_t, the Sampling Theorem requires I to be band-limited, so that its Fourier transform vanishes beyond $\pm\frac{1}{2|v|}$. This can be achieved by smoothing with a spatial Gaussian. However, smoothing poses a notable drawback, as it distorts the image data, causing features to disappear, merge and blur.

An alternative approach, closely related to the concept of "warping" (e.g., [17]), would be to take advantage of prior estimates of the optical flow. If a point is estimated to move at velocity $u = (u_x, u_y)$, we can use the convolution of I in the direction of $(u_x, u_y, 1)$ to estimate the directional derivative I_u and apply

$$I_t = I_u - u_x I_x - u_y I_y \tag{35}$$

The convolution that yields I_u is equivalent to subsampling in the direction of $(v - u)$, and thus the estimate of I_t is unaliased if the Fourier transform vanishes beyond $\pm\frac{1}{2|v-u|}$. This occurs when either the estimated velocity u is close to the real velocity v, or the region is smooth. This is particularly important, as the estimation of optical flow in smooth regions is often inaccurate. In other words, this estimation of I_t is tolerant to inaccuracy in motion estimation exactly when it is least reliable. The figures in Secion 3 demonstrate our algorithm's tolerance to poor motion estimation.

Note that the spatial smoothness of u is not required. Also note that temporal smoothing has no effect on the aliasing problem, and it is desirable to have as little temporal smoothing as possible.

II.2. Differentiation with Two Frames

Differentiation, as described earlier, is done by convolution with derivatives of a spatio-temporal Gaussian, which requires several frames to achieve a good estimation. When only two frames are available, special care should be taken to provide a consistent estimation of spatial and temporal derivatives. Given two frames $I(x, y, 0)$ and $I(x, y, 1)$, let us define

$$I^*(x, y, t) = \begin{cases} I(x, y, 0) & t \le 0 \\ I(x, y, 1) & t > 0 \end{cases} \tag{36}$$

Then, for any temporal variance s_t, the partial derivative estimates are

$$\begin{aligned} I_x^* &= \frac{1}{2}(I(x, y, 0) + I(x, y, 1)) * g_x \\ I_y^* &= \frac{1}{2}(I(x, y, 0) + I(x, y, 1)) * g_y \\ I_t^* &= (I(x, y, 1) - I(x, y, 0)) * g \end{aligned} \tag{37}$$

(where $*g$, $*g_x$ and $*g_y$ denote convolutions with the spatial Gaussian and with its X and Y derivative respectively).

II.3. Application to Optical Flow

It is well known that the computation of optical flow in textureless regions and along straight lines (aperture problem) is ill-posed. When these situations occur, the rank of $\mathbf{G}$ is 0 and 1, respectively. These situations arise from spatial structure alone, and can therefore be detected by the spatial 2D second moment matrix (used, for example, in the Lucas-Kanade algorithm [17]), in order to mark these regions as unreliable (as done in many implementations). Optical flow is also unreliable at motion boundaries, which may be treated by the joint estimation of motion and segmentation [28], [33].

These two cases can be treated jointly using the rank of $\mathbf{G}$. Optical flow in regions where $rank(\mathbf{G}) \neq \mathbf{2}$ can be estimated by filling from adjacent regions where $rank(\mathbf{G}) = \mathbf{2}$. In a coarse-to-fine algorithm, this should be done at each scale.

ACKNOWLEDGMENT

This research was supported by the EU under the DIRAC integrated project IST-027787.

REFERENCES

[1] N. Apostoloff, A. Fitzgibbon. Learning spatiotemporal T-junctions for occlusion detection. *CVPR'05*, 553–559.
[2] L. Bergen and F. Meyer. A novel approach to depth ordering in monocular image sequences. *CVPR'00*, II:536–541.
[3] M.J. Black, D.J. Fleet. Probabilistic Detection and Tracking of Motion Boundaries. *IJCV* 38(3):231–245, 2000.
[4] G. T. Chou. A Model of Figure-Ground Segregation from Kinetic Occlusion. *ICCV'95*, 1050–1057.
[5] A. W. Cunningham, T. F. Shipley, P. J. Kellman. Interactions between spatial and spatiotemporal information in spatiotemporal boundary formation. *Percept. Psychophys.* 60(5):839-851, 1998.
[6] T. Darrell and D.Fleet. Second-order method for occlusion relationships in motion layers. MIT Media Lab Technical Report 314, 1995.
[7] D. Feldman and D. Weinshall. Motion Segmentation Using an Occlusion Detector. In *Proc. of Workshop on Dynamical Vision, ECCV'06*, May 2006.
[8] E. Gamble, T. Poggio. Visual integration and detection of discontinuities: The key role of intensity edges. AI Lab, MIT, A.I. Memo No. 970, 1987.
[9] C. Harris, M. Stephens. A combined corner and edge detector. In *Proc. of Alvey Vision Conference*, 147–151, 1988.
[10] B. Horn, B.Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.
[11] G. A. Kaplan. Kinetic disruption of optical texture: the perception of depth at an edge. *Percept. Psychophys.* 6(4):193–198, 1969.
[12] Q. Ke, T. Kanade. A subspace approach to layer extraction. In *Proc. of CVPR'01*, 255–262.
[13] V. Kolmogorov, R. Zabih. Computing Visual Correspondence with Occlusions using Graph Cuts. In *Proc. of ICCV'01*, 2:508–515.
[14] I. Laptev, T. Lindeberg. Space-time Interest Points. In *Proc. of ICCV'03*, 432–439, 2003.
[15] I. Laptev, T. Lindeberg. Velocity adaption of space-time interest points *ICPR'04*, 1:52-56.
[16] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. *IJCV*, 30(2):117–154, 1998.
[17] B. D. Lucas, T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proc. of IJCAI'81*, 674–679, 1981.
[18] M. Middendorf, H.-H. Nagel. Estimation and Interpretation of Discontinuities in Optical Flow Fields. In *Proc. of ICCV'01*, 1:178-183.
[19] D. W. Murray, B. F. Buxton. Scene segmentation from visual motion using global optimization. *TPAMI*, 9(2):220–228, March 1987.
[20] K. M. Mutch, W. B. Thompson. Analysis of Accretion and Deletion at Boundaries in Dynamic Scenes. *TPAMI* 7(2):133–138, 1985.
[21] M. Nicolescu, G. Medioni. A Voting-Based Computational Framework for Visual Motion Analysis and Interpretation. *TPAMI*, 27(5):739–752, May 2005.

[22] S. A. Niyogi. Detecting kinetic occlusion. *ICCV'95*, 1044–1049.
[23] J.M. Odobez, P. Bouthemy. MRF-based motion segmentation exploiting a 2D motion model robust estimation. In *Proc. of ICIP'95*, 3:628–631, 1995.
[24] A. S. Ogale, C. Fermüller, Y. Aloimonos. Motion Segmentation Using Occlusions. *TPAMI* 27(6):988–992, June 2005.
[25] H. Pao, D. Geiger, N. Rubin. Measuring Convexity for Figure/Ground Separation. *ICCV'99*, 948–955.
[26] X. Ren, C. C. Fowlkes, J. Malik. Figure/Ground Assignment in Natural Images. *ECCV'06*, II:614–627.
[27] E. Saund. Perceptual Organization of Occluding Contours of Opaque Surfaces *CVIU*, 76(1):70–82, October 1999.
[28] H. S. Sawhney, S. Ayer. Compact Representations of Videos Through Dominant and Multiple Motion Estimation. *TPAMI*, 18(8):814–830, August 1996.
[29] J. Shi, J. Malik. Motion Segmentation and Tracking Using Normalized Cuts. In *Proc. of ICCV'98*, 1154–1160, 1998.
[30] M. Tappen, W. T. Freeman. Comparison of Graph Cuts with Belief Propagation for Stereo, using Identical MRF Parameters. In *Proc. of ICCV'03*, 900–907, 2003.
[31] W. B. Thompson, K. M. Mutch, V. A. Berzins. Dynamic Occlusion Analysis in Optical Flow Fields. *TPAMI* 7(4):374–383, 1985.
[32] D. Tweed, A. Calway. Integrated Segmentation and Depth Ordering of Motion Layers in Image Sequences. *BMVC'00*.
[33] Y. Weiss. Smoothness in layers: motion segmentation estimation using nonparametric mixture. In *Proc. of CVPR'97*, 520–526, 1997.
[34] Y. Weiss, E. H. Adelson A unified mixture framework for motion segmentation: incorporating spatial coherence and estimating the number of models. In *Proc. CVPR'96*, 321–326.
[35] J. Xiao, M. Shah Accurate Motion Layer Segmentation and Matting In *Proc. CVPR'05*, 698–703.
[36] A. Yonas, L. G. Craton, W. B. Thompson. Relative motion: Kinetic information for the order of depth at an edge. *Percept. Psychophys.* 41(1):53–59, 1987.