https://doi.org/10.2352/ISSN.2470-1173.2017.14.HVEI-137

A Neurally-Inspired Algorithm for Detecting Ordinal Depth from Motion Signals in Video Streams

Gennady Livitz, Harald Ruda, & Ennio Mingolla
Computational Vision Laboratory, Northeastern University, Boston, Massachusetts

Abstract

Understanding the depth order of surfaces in the natural world is one of the most fundamental operations of the visual systems of many species. Humans reliably perceive the depth order of visually adjacent surfaces when there is relative motion between them such that one surface appears or disappears behind another. We have adapted a computational model of primate vision that fits important classical and recent psychophysical data on ordinal depth from motion in order to develop a fast, robust, and reliable algorithm for determining the depth order of regions in natural scene video. The algorithm uses dense optic flow to delineate moving surfaces and their relative depth order with respect to the parts of the static environment. The algorithm categorizes surfaces according to whether they are emerging, disappearing, unoccluded, or doubly occluded. We have tested this algorithm on real video where pedestrians and cars sometimes go behind and sometimes in front of trees. Because the algorithm extracts surfaces and labels their depth order, it is suitable as a low-level pre-processing step for complex surveillance applications. Our implementation of the algorithm uses the open source HPE Cognitive Computing Toolkit and can be scaled to very large video streams.

Introduction: Depth Order from Motion

Biological visual systems effectively use motion cues for scene segmentation. For a prey with effective static camouflage, the change in optical signals resulting from their motion may be the only cues available to a predator's visual system for detection surface segregation (of prey from background or emerging from behind an obstacle). The motion of one surface behind another, illustrated in Figure 1, provides powerful and reliable cues for depth order (but *not* metric depth). These *accretion* and *deletion* cues are always associated with the background surface. Consider a moving background behind a static figure, or a moving background seen through an aperture. The accretion and deletion cues are just as powerful and useful in natural video. As will be discussed in the description of our algorithm, the notion of *accretion* or *deletion* of discrete featural elements (e.g., dots or texture) can be generalized to growth or erosion of relatively homogeneous image regions.

Inspired by models of primate vision that explain depth order based solely on kinetic clues [1][2][3], we have created the CogMO (Cog-Motion-Occlusion) algorithm. The essential insights of previous models, that the form and motion pathways of the primate visual system are mostly separated but must interact, and that there exist simple mechanisms for detecting accretion and deletion, have been incorporated into the CogMO algorithm.

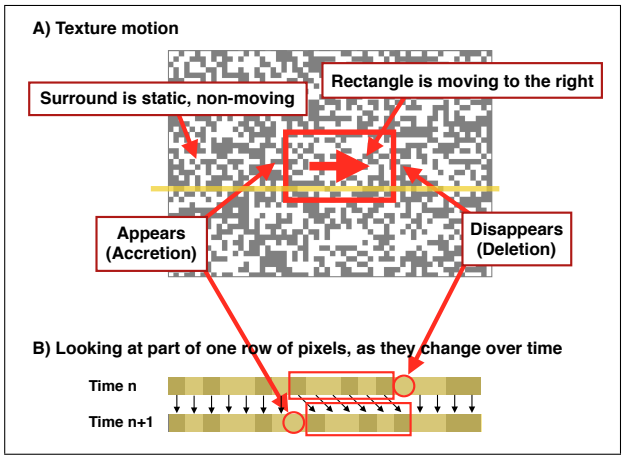

Figure 1: When a random noise rectangle moves to the right in front of a random noise background (A), the appearance and disappearance of dots by the sides of the rectangle (*accretion* and *deletion* cues) creates a clear depth order – the moving rectangle is seen in front of the surround. (B) Examining more closely one row of hypothetical pixels, accretion and deletion can be detected when a pixel has no corresponding counterpart in the previous or next frame (marked with circles).

CogMO Overview

The CogMO algorithm described here uses only motion signals to derive depth order from video. This algorithm segregates isolated moving surfaces from static background and establishes depth relations (such as "occludes", or "being occluded

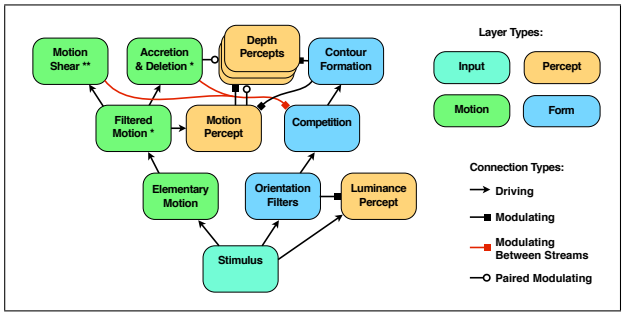

Figure 2: Diagram of the algorithm in [1] that is the main inspiration for the current work. Note the separate *where* (motion processing) and *what* (form processing) streams.

by") along the boundaries of moving surfaces. Note that this is *not* a "3D reconstruction", but a "layering" of the environment using motion cues.

Furthermore, inferring depth relationship in the dynamic scene allows our algorithm, over time, to learn the location and the shape of static occluding objects and potentially complete occluded surfaces "amodally". This means that an (incomplete) representation of the whole object (e.g., a bounding box) can be inferred without reliance on any domain knowledge (such as from a classifier) about the identity of the object undergoing occlusion.

In order to produce motion signals, we used the Zach et al. dense optic flow algorithm [4]. Our CogMO algorithm performs in near real time[1] (2 fps) on video recorded of a street scene. The moving objects in the scenes (cars, trucks, pedestrians) are sometimes occluded by trees. Our algorithm in most cases correctly identifies ordinal depth in the scene and learns locations of the occluding trees. These relationships are marked in real time for human visualization using color and luminance.

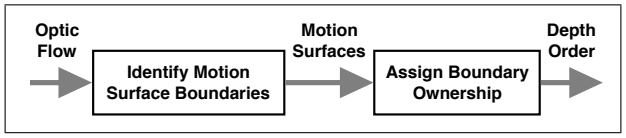

Figure 3: Overview of the CogMO algorithm.

Our algorithm makes its depth order assessment entirely on the basis of motion signals determined from the optical flow field. This allows to isolate motion processing as a module that can be used as a component in a larger more comprehensive vision system. For more robust scene interpretation, this algorithm can be combined with other algorithms that exploit luminance and color (see for example [5]).

The objectives of the proposed algorithm are to identify moving surfaces and for each moving surface to classify its boundaries as belonging either to the surface itself or to the static foreground that partially occludes a given moving surface. Making such an inference is equivalent to establishing ordinal depth relationship for certain pairs of adjacent surfaces in a scene. For an explanation of the importance of surfaces in the context of human motion perception see [6]. The functional control flow of the CogMO algorithm is shown in Figure 3.

Computational Environment

In order to process natural video on a realistic time scale, we implemented our algorithm in a special computational environment, *Cog Ex Machina* (CogX, version 4) [7]. CogX uses and abstracts available computational resources (GPUs) and enforces parallel computation through a special modeling language. This language supports formulation of the algorithm in terms of algebraic manipulation of tensor fields (multidimensional arrays) that are used to represent abstractions of computations that can be viewed as done by "neural units".

CogMO Algorithm

To determine ordinal depth relationships, the visual image undergoes a sequence of transformations; each of these has an analogy in primate visual processing. Figure 4 shows the data flow and relationship between functional units of the proposed algorithm. The rest of this section is a description of each of the functional modules shown in the figure. In what follows, "box 1", "box 2", etc. refers to the labels in the top left corner of the boxes in this figure.

Optic Flow

Optic flow (box 2, Figure 4) refers to a velocity field (two-component vector field $O(0)$ and $O(1)$) generated by processing a stream of video $V[t]$ (box 1, Figure 4), where V is a luminance scalar field. Optic flow algorithms used in visual image processing can be largely divided into two classes. Dense algorithms produce velocities for all locations in the field. Sparse algorithms produce velocities only for specified locations. While sparse algorithms require fewer computational resources and presumably rely on less noisy data, they are concerned with a higher-level correspondence problem. For our low-level algorithm, dense optic flow is required.

Discretized Motion

The representation used for optic flow is an analog vector. The CogMO algorithm works with a discrete set of motion directions. Discrete representations are generally less noisy than analog representations. The operation which transforms optic flow into discrete motion representation (box 3, Figure 4) is specified by Equations 1–4. Motion direction is encoded by a normal vector $m(d)$, where $d = 0, 1...n-1$ that span 2D direction space with n directions and intervals of size $360°/n$. The directional angle ϕ is computed from the optic flow according to:

$$\phi = \text{atan2}(O(0), O(1)) \tag{1}$$

where function $atan2()$ uses the inverse tangent ($arctan$) function to find an angle in $[-\pi, -\pi]$ (i.e., full 360°) interval specified by

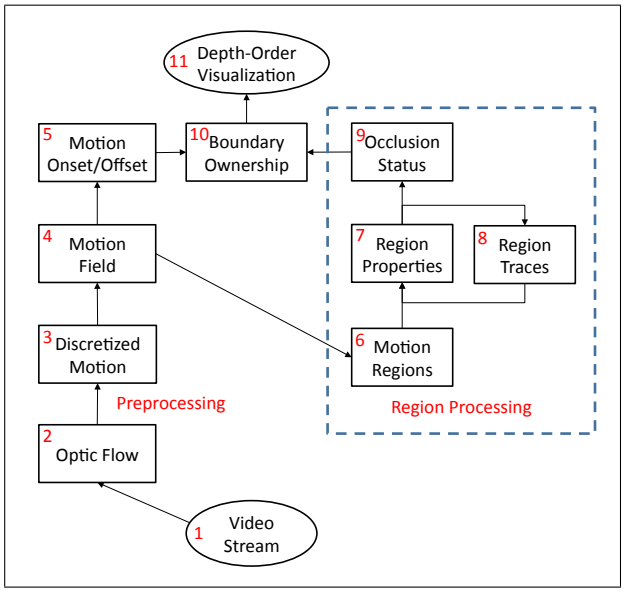

Figure 4: Detailed CogMO algorithm data flow and component interaction diagram.

[1] HP Z820 workstation with an NVIDIA GeForce GTX TITAN GPU

two vector components. The motion direction D is found as the closest vector to m according to Equations 2 and 3:

$$\phi_{min} = \min(|\phi - m(d)|) \text{ for } d = 0...n-1, \quad \text{where } m(d) = \text{atan2}(2\pi d/n) \tag{2}$$

$$D(d) = \begin{cases} 1 & \text{if } |\phi - m(d)| = \phi_{min} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Direction signals are thresholded to eliminate motion noise to produce a discretized motion signal M according to Equation 4,

$$M = D(\max(O) > G_m)[\|O\| - L_m]^+ \tag{4}$$

where L_m is a local motion magnitude threshold and G_m is a global magnitude threshold. Thus, a discretized motion representation M is a vector field that approximately preserves velocity direction. The number of vector components in such a field is equivalent to the number of directions of velocity (n), but at most one tensor component can be non-zero.

Motion Field

The motion field (box 4, Figure 4) S is a binary scalar field encoding location where discretized motion is present. This representation is an intermediate step required to enumerate motion surfaces — connected regions of coherent motion. Determining locations where a discretized motion vector field is non-zero is done by selecting a non-zero component of the discretized vector field M according to Equation 5,

$$S = M_{max}. \tag{5}$$

Here M_{max} selects maximal component of each vector of the vector field M. This operation constitutes tensor rank reduction from 1 to 0 (i.e., vector to scalar). Examples of motion fields are shown in the second row of Figure 5.

Motion Boundaries, Motion Onset and Offset

Motion boundaries (box 5, Figure 4) delineate the extent of a region of motion. A location on the motion boundary can be labeled as motion onset or motion offset. Onset of motion encodes locations where the motion emerges and the offset encodes locations where the motion disappears. This determination is possible only by analyzing the optic flow signals. The motion boundary signal B is computed according to

$$B(d) = S - S^{\gg m(d)}, \tag{6}$$

where $S^{\gg d}$ designates the shift of the whole image S by a directional vector $m(d)$; $m(d)$ encodes a directional vector for the direction $2\pi d/n$. Oriented motion onset B^+ and offset B^- signals are computed according to Equations 7 and 8:

$$B^+(d) = M(d)[M(d) - M^{\gg m(d)}]^+ \tag{7}$$

$$B^-(d) = M(d)[M(d) - M^{\gg m(\text{comp}(d))}]^+ \tag{8}$$

Note that $comp(d)$ represents the direction opposite to $m(d)$. Motion directions orthogonal to the direction of the boundary are suppressed as they do not represent motion onset/offset, in Equations 9 and 10:

$$B^+(d) = B^+(d)[B(\text{ort}(d)) + B(\text{comp}(\text{ort}(d)), 1] \tag{9}$$

$$B^-(d) = B^-(d)[B(\text{ort}(d)) + B(\text{comp}(\text{ort}(d)), 1] \tag{10}$$

Where d corresponds to one of the n basic directions spanning direction space with $360°/n$ intervals; $ort(d)$ represents direction perpendicular to d (in the sense of $+90°$). Finally, scalar representation for motion onset and motion offset is computed by tensor reduction:

$$B^{\pm} = B^{\pm}_{max} \tag{11}$$

Detecting Motion Surfaces

Motion surfaces (box 6, Figure 4) are formed by connected and coherently moving pixels. Some of these surfaces may belong to the same objects while some of them may not. The reasons for discontinuities within a single object can be occlusions or noise resulting from optical flow computation. The reason for any given discontinuity can be identified by a proper motion field analysis. Disconnected surfaces potentially can be grouped together as parts of the same object. However at this stage of visual image processing, the goal is to delineate motion surfaces and to process surfaces instead of pixels. Motion surfaces represent a higher layer in the hierarchy of visual data encoding than pixel representation. Unlike pixels, which can be characterized only

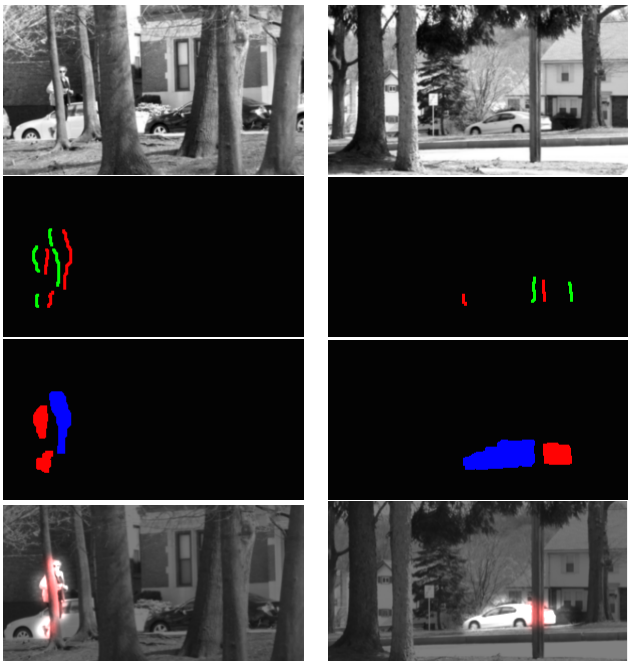

Figure 5: The top row shows the input frames to the algorithm chosen from processing of two video sequences. On the left side a pedestrian is walking behind a tree (the cars are parked). On the right side, the car is moving to the left. The second row show the motion onsets (red) and offsets (green). The third row shows the moving regions labeled by the algorithm as emerging (blue) and disappearing (red). The bottom row shows the moving regions (lightened) together with the actual occluders (reddish).

by their luminance value, motion surfaces can be described by a multidimensional set of properties characterizing their shape and dynamics. However motion surface properties are uniform, associated with the entire region and therefore significantly reduce the bandwidth required for their description and communication. The motion surfaces consist of individual pixels, but a pixel representation of the scene already exists at the earlier stages of visual processing described above (boxes 2–4, Figure 4) and once motion surface boundaries are determined they can be used to extract pixel representation of a region.

We use an iterative recurrent dynamic filling-in process (still box 6, Figure 4) to segregate individual motion surfaces from the motion field S. In the process of surface demarcation, an initial seed placed within a connected region is allowed to spread by convolving the seed with a small (3x3) rectangular kernel with subsequent gating by the original motion field and thresholding the results at the end of each iteration. The process is continued until the resulting field stops changing. At this point, the region is considered delineated, assigned to a corresponding layer and subtracted from the original motion field in order to identify the next seed. The process continues until no seed can be placed within the original motion field, because all its non-zero locations would belong to identified motion surface.

Filling-in processes were serialized in order to enumerate individual surfaces. The details of surface enumeration are specified by Equations 12 and 13:

$$g(i) = \begin{cases} S, & i = 0 \\ \overline{g(i-1) * C(i-1)}, & i > 0 \end{cases} \tag{12}$$

$$C(i) = S * \text{spread}(\text{wta}(g(i)), g(i))|_i \quad i = 0, 1, \ldots \sum_{k,l} g_{kl}(i) > 0 \tag{13}$$

Where, k is a contour index, and $\bar{x}$ is defined by

$$\bar{x} = \begin{cases} 0 & x > 0 \\ 1 & x = 0 \end{cases} \tag{14}$$

$C(i)$ is a motion field, $g(i)$ represents the ith iteration in a process of surface finding $d = spread(s, f)$ is an operator that finds subset of pixels connected to pixel s specified as a seed in a field f. It performs a set of subsequent convolutions with a small (3x3) rectangular kernel K

$$d(i) = \begin{cases} |\text{conv}(s,K), T_K|^+ * f, & i = 0 \\ |\text{conv}(d(i-1),K), T_K|^+ * f, & i > 0 \end{cases} \quad \text{for } i = 0, 1, \ldots d(i) \neq d(i-1) \tag{15}$$

where

$$\text{conv}(x, K) = \sum_{ij} \sum_{kl} x_{i-k+1, j-l+1} * K_{ij} \tag{16}$$

and $wta(x)$ suppresses all pixels in the field x except the ones that have the maximum value.

$$\text{wta}(x) = \begin{cases} 1, & x_{ij} \geq x, \forall x \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

The results of this process are multiple fields; examples are shown in the third row of Figure 5. Thus final representation of image as a collection of motion surfaces is a vector field where the number of vector components is equivalent to a number of motion surfaces

The following motion surfaces properties (box 7, Figure 4) are relevant to determination of boundary ownership: centroid location, $U(i)$ and velocity, $v(i)$. These properties are computed as following:

$$U(i) = \left(\frac{\sum_{k,l, C_{kl} \neq 0} k}{R_i}, \frac{\sum_{k,l, C_{kl} \neq 0} l}{R_i} \right) \tag{18}$$

$$v(i,d) = \sum_{k,l} C_{kl}(i) M(d) \tag{19}$$

$$v(i) = \sum_{d} (m(d) * (max(O(i,d)) == O(i,d))) \tag{20}$$

Here, i represents a motion surface index; d represent a discrete motion direction, M is a discretized motion signal; $m(d)$ corresponds to a discrete velocity for direction d. The motion onset and offset boundaries are described by:

$$B^+(i) = \frac{\sum_{k,l, C_{kl} B^+ \neq 0} C(i) B^+}{R_i} \tag{21}$$

$$B^-(i) = \frac{\sum_{k,l, C_{kl} B^- \neq 0} C(i) B^-}{R_i} \tag{22}$$

Region Dynamics

Regions dynamics (box 8, Figure 4) provides important cues in determination of ordinal depth of a motion surface. This process includes finding surface correspondence between two frames. Finding correspondence is important because incorrect correspondence would lead to incorrect judgment about the state of occlusion.

The algorithm operates on the assumption that surface locations overlap in two consecutive frames. If they do not, the surface has just appeared from occlusion. This assumption holds for most of the real situations if sampling happens with sufficient frequency. Therefore the surface enumeration signal $C(i)$ related to the previous frame is preserved.

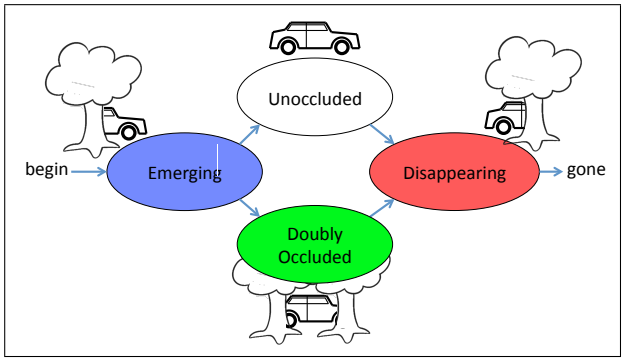

Figure 6: Occlusion states and ideal state transitions. The car is occluded by one or more trees. The CogMO system allows for a noisy signal as transitions between states only occur when there is sufficient confidence.

Once surfaces for the current frame are found, the algorithm processes them sequentially, determining the ancestor of each surface (i.e., the same surface in the previous frame). This can only be done by knowing the properties and dynamics of all the possible ancestors of the surface. The following surface properties are found by using surface dynamics and surface boundary dynamics:

1. Overlap of the contour i with contours in the previous frame

$$I(i) = \sum_i C(i) * C(i)[t-1] \tag{23}$$

2. Number of overlaps

$$N(i) = \sum_{i, I(i) \neq 0} 1 \tag{24}$$

3. The location of the motion onset and offset boundaries

$$B_c^+(i) = \frac{\sum_{i, I(i) \neq 0} B^+(i)}{N(i)} \tag{25}$$

$$B_c^-(i) = \frac{\sum_{i, I(i) \neq 0} B^-(i)}{N(i)} \tag{26}$$

Note that the $B^+(i)$ here from Equation 21 is different from the $B^+(d)$ of Equation 7.

4. Boundary displacements

$$\Delta B^+(i) = B^+(i) - B_c^+(i) \tag{27}$$

$$\Delta B^-(i) = B^-(i) - B_c^-(i) \tag{28}$$

5. Extent of the region

$$\Delta B(i) = |B^+(i) - B^-(i)| \tag{29}$$

Dynamics of Occlusions

A surface region can be in one of the following four states with respect to occlusion dynamics (see Figure 6):

1. Unoccluded (O^U): no surface occludes the region.
2. Disappearing (O^D): the surface is becoming more occluded and partially hidden behind an occluding surface. The occluding surface is located in the direction pointed by the region's velocity vector $o(i)$ computed according Equation 20.
3. Emerging (O^E): the surface is coming out of occlusion but still partially hidden behind an occluding surface. The occluding surface is located in the direction opposite to the region's velocity vector $o(i)$.
4. Doubly-Occluded (O^B): in this state the surface is located between occluders. The occluding objects are located in the direction indicated by the region's velocity vector $o(i)$ and also in the opposite direction.

The occlusion state can be determined (box 9, Figure 4) with a various degree of reliability from a number of cues. We implemented a scoring system that takes into account a dynamic of motion onset/offset boundaries B^+B^- and extent of the region ΔB. In general, a moving boundary corresponds to an unoccluded moving edge, while a static boundary corresponds to an occluding edge belonging to a static foreground. Displacement of the unoccluded edge towards the static occluding edge correlates to the "Dissapearing" state (O^D), while displacement of the unoccluded edge away from static boundary corresponds to the "Emerging" state (O^E). The sign of displacement differentiates emergence from disappearance. The size of displacement correlates with the probability of a region being occluded from one side versus "Unoccluded" or "Doubly-Occluded" state. The Unoccluded state correlates with the total displacement of both boundaries or a region's centroid, while the double occluded state can be characterized by a lack of boundary motion. Thus, we can introduce a scoring system, which would allow us to select the region's status based on regions boundary dynamics.

Real time motion signals from video are notoriously noisy, and therefore any of a surface's properties can be a result of error, random fluctuation, some weird shape irregularity, etc. To make our algorithm more robust, in addition to the scoring system based on the dynamics of region's boundaries, we added a "state transition inertia", which enables a state transition only after certain stability threshold in assigning occlusion status to a given region is reached.

$$o^D(i) = |\Delta B^-(i)| - |\Delta B^+(i)| \tag{30}$$

$$o^E(i) = -o^D(i) \tag{31}$$

$$o^U(i) = \frac{|\Delta B^-(i)| + |\Delta B^+(i)|}{2} \tag{32}$$

$$o^B(i) = v(i) - o^U(i) \tag{33}$$

And using

$$o_{max}(i) = \max(o^D(i), o^E(i), o^U(i), o^B(i)) \tag{34}$$

We then define $O^D(i), O^E(i), O^U(i), O^B(i)$ according to:

$$O^x(i) = \begin{cases} 1 & \text{if } o^x(i) = o_{max}(i) \\ 0 & \text{otherwise} \end{cases} \quad \text{for } x \in \{D, E, U, B\} \tag{35}$$

Here o^D, o^E, o^B, and o^U represent a score of the corresponding region state. After these are computed, the maximum is chosen and only the state of (O^D, O^E, O^B, and O^U) is set for the region. Thus the occlusion state is the result of evaluating a surface's boundary dynamics and also taking into account state transition inertia.

Boundary Ownership

Motion onsets and offsets B^+ and B^- can be qualified (box 10, Figure 4) as a moving surface onset H^+, moving surface offset H^-, a static surface occluding edge P^- and static surface disoccluding edge P^+ (see second row of Figure 5).

$$H^+(i) = C(i) * B^+(i) * (O^E(i) | O^U(i)) \tag{36}$$

$$H^-(i) = C(i) * B^-(i) * (O^D(i) | O^U(i)) \tag{37}$$

$$P^+(i) = C(i) * B^+(i) * (O^D(i) | O^B(i)) \tag{38}$$

$$P^-(i) = C(i) * B^-(i) * (O^E(i) | O^B(i)) \tag{39}$$

Here $C(i)$ has all locations that belong region I, set to 1, B^+ defines region onset, and B^- defines region offset. O^D, O^E, O^B, and O^U flags specify motion region's status.

To be an edge of a moving surface H (Equations 36–37), the boundary should belong to an occluding (O^E) region and be a motion region onset or belong to a disoccluding (O^D) region and be a motion offset or belong to an unoccluded region (O^U). To be an edge of a static occluding surface (Equations 38–39), the boundary should belong to either occluding (O^D) or disoccluding (O^E) or a double occluded (O^B) region and be a motion surface offset in case of O^E or a motion surface onset in case of O^D. The latter requirement reflects motion disappearance and motion emergence at the edges of occluding surface.

For future reference, we define a moving surface region's *front edge* (H^+) and *rear edge* (H^-) with respect to direction of velocity O computed for the region. Similarly, for static object edges, let's refer the edge where motion emerges the *front edge* and where it disappears the *rear edge*.

Depth Relations

The CogMO algorithm described in this document derives depth relations (box 11, Figure 4) from the boundary ownership signals: H and P. These relations are local, in the sense that they exist in the vicinity of motion boundaries. The following depth relationship can be derived from boundary ownership H and P and region velocity $o(i)$:

The area in the vicinity of the occluding surface's front boundary P^+, in the direction pointed by the region's velocity vector $o(i)$ is behind the area pointed to by the opposite of the surface velocity direction.

The area in the vicinity of the occluding surface's rear boundary P^-, in the direction pointed by the region's velocity vector is in front of the occluding surface region to which rear boundary P^- belongs.

The area in the vicinity of the moving surface's front boundary H^+ in the direction pointed by the region's velocity vector $o(i)$ is behind the is behind the area pointed by the opposite to the region velocity direction.

The area in the vicinity of the moving surface's front boundary H^- in the direction pointed by the region's velocity vector $o(i)$ is in front of the occluding surface region to which rear boundary H^- belongs.

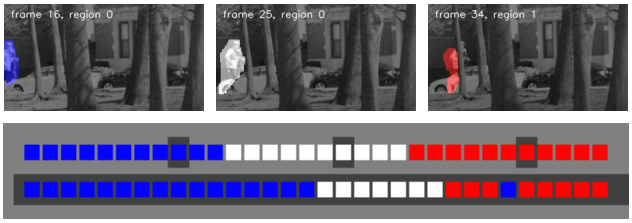

Figure 7: The top three frames are from the first event of the pedestrian video. The highlighted portion in each frame is the region found by the CogMO algorithm; it is highlighted according to the ground truth established by careful human observation. The first frame shows the pedestrian emerging from 'behind' the left edge of the frame; the second frame shows the unoccluded pedestrian; and the third frame shows the pedestrian disappearing behind a tree. The bottom part of the figure shows one column for each unique frame and region for this event. The colors correspond to the colors and states shown in Figure 6, blue is Emerging, white is Unoccluded, red is Disappearing, and green is Doubly Occluded. The top row is Ground Truth, with the frames shown above marked with a black border. The bottom row shows the CogMO output.

Visualizing Ordinal Depth

A useful way to represent the depth relations inferred according to the rules specified in the previous section is to use three distinct visualizers:

- Moving surfaces – INSIDE (I)
- Static occluders – NEAR (N)
- The rest of the scene – FAR (F)

This approach allows visualization of up to 3 depths strictly based on local relationship by applying the rules stated in the previous section. These rules are formally applied according to

$$N(i) = \sum_{j=0\ldots n-1} \text{warp}(P^+(i), j*O(i)) + \sum_{j=0\ldots n-1} \text{warp}(P^-(i), -j*O(i)) \tag{40}$$

$$I(i) = \text{conv}(C(i), \text{gaus}(2.0)) \tag{41}$$

$$F = 1 - \sum N(i) - \sum F(i) \tag{42}$$

Where i identifies a region, F represents the scene background, N represents the area occluding a moving surface (static occluders), I represents an unoccluded moving surface, j represents an iteration in the process of depth propagation in the vicinity of occluding edges $warp(a,b)$ – an operator that shifts a field in the direction pointed by vector b n – number of iterations in depth propagation (visualization parameter $n = 10$) $\text{conv}(a,k)$ – convolves a scalar field a with a kernel k, $\text{gauss}(w)$ – Gaussian kernel defined as $(1/2\pi\sigma)exp(-d^2/2\sigma^2)$; $2w$ is a kernel width, d is a distance in pixels to kernel center.

The bottom row of Figure 5 shows visualisation of depth order for the frames displayed in the top row. The moving regions (INSIDE) are highlighted, the static background (FAR) is dimmed, while any occluding areas (NEAR) are shown reddish.

Evaluation

We compared the performance of the algorithm with that of human observers (Figures 7 and 8). We were interested in human performance in real-time, using just a few frames to decide the occlusion state of a selected region. Thus we created a simple psychophysical task using PsychToolbox [8] where for each region created by the CogMO algorithm, three frames were presented in rapid succession to enable the sensation of movement. The selected region was highlighted (as in the middle frame of the top row of Figure 7). Observers were presented with every three-frame sequence in a random order and asked about the occlusion state (E,D,U,B) of the highlighted region. It is necessary to highlight the surface being considered, as sometimes a pedestrian would be visible on both sides of a tree, and thus simultaneously Emerging and Disappearing.

To evaluate the performance of human observers and of the CogMO algorithm, a "Ground-Truth" was established by careful

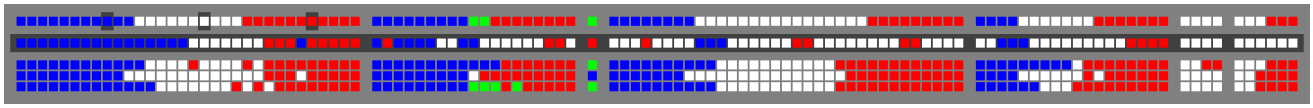

Figure 8: The figure shows one column for each unique frame and region, divided into events. The colors correspond to the colors and states shown in Figures 6 and 7. The rows are (in order from top to bottom) Ground-Truth, CogMO, then three human observers (GL, AR, HR). The three frames shown in the top part of Figure 7 are again highlighted with a black border in the "Ground-Truth" row.

back-and-forth stepping through frames. The main criterion used to determine occlusion of a pedestrian was that the torso or head had to be occluded; a foot appearing or disappearing behind a tree is not sufficient.

The results of the comparison is presented in Table 1. The performance of the CogMO algorithm (58%, using the *loose* standard for Doubly Occluded, *B*) is not on par with human observers (87%). Interestingly, the performance of human observers when compared to other observers is slightly lower (85%), indicating a fairly high variability among human observers.

The CogMO algorithm, while better than random (33%), will occasionally make incongruous judgments, such as the one frame labeled *E* among the *D* frames of sequence one. The CogMO algorithm is more likely than human observers to categorize surfaces as Unoccluded. While it is possible to add mechanisms whereby the CogMO algorithm can be tuned, please note that as specified herein, it has no tunable (free) parameters.

Performance	vs GT	vs Os	vs GT	vs Os
CogMO	56.6%	54.0%	58.4%	54.9%
Observers	85.5%	82.6%	87.3%	85.2%
	strict *B*		loose *B*	

Table 1: The performance of the CogMO algorithm. GT refers to "Ground-Truth" established by careful frame-by-frame observation. Os simply means Observers. The performance of both the CogMO algorithm and Observers are compared both the Ground Truth and Observers. For the Observers statistics, they are averaged over all Observers, and when compared to Observers that means *other* Observers. The *strict* and *loose B* columns refer to the interpretation of the Doubly Occluded state. When *strict*, a judgement of *B* has to match a Ground Truth of *B*. When *loose*, a judgment of *B* can match a Ground Truth of *E*, *B*, or *D*; and vice versa.

Summary

The algorithm described in this paper processes natural video in close to real-time and is able to extract moving regions. In addition, because video is a source of dynamic information, the algorithm is able to classify the "occlusion state" of each region for each frame. Both determining the occlusion state and comparing with human performance on this task appears to be a completely novel procedure. While the performance of the algorithm is adequate, it does not reach human levels, suggesting that the human visual system may make use of additional types of processing.

The CogMO algorithm is useful because the notion of occlusion status of a surface can be used to keep track of objects in the environment. For example, in an automated surveillance scenario, it can be used to recognize the difference between going out of frame vs hiding behind an object in the camera field of view.

Acknowledgments

This work was supported in part by a subcontract to Northeastern University from Hewlett-Packard Enterprise: OAK Sentient Prime Contract # 15-C-0249.

References

[1] Harald Ruda, Gennady Livitz, Guillaume Riesen, and Ennio Mingolla. Computational modeling of depth ordering in occlusion through accretion or deletion of texture. *Journal of Vision*, 15(9):20, 2015.

[2] Timothy Barnes and Ennio Mingolla. A neural model of visual figure-ground segregation from kinetic occlusion. *Neural Networks*, 37:141–164, 2013.

[3] Timothy Barnes and Ennio Mingolla. Representation of motion onset and offset in an augmented barlow-levick model of motion detection. *Journal of computational neuroscience*, 33(3):421–434, 2012.

[4] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Pattern Recognition*, pages 214–223. Springer, 2007.

[5] Patrik Sundberg, Thomas Brox, Michael Maire, Pablo Arbelaez, and Jitendra Malik. Occlusion boundary detection and figure/ground assignment from optical flow. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2233–2240. IEEE, 2011.

[6] Heiko Neumann, Arash Yazdanbakhsh, and Ennio Mingolla. Seeing surfaces: The brain's vision of the world. *Physics of Life Reviews*, 4(3):189–222, 2007.

[7] Greg Snider. Massively parallel computing on cog ex machina. *HP Laboratories Technical Report*, 179:1–11, 2012.

[8] Mario Kleiner, David Brainard, Denis Pelli, Allen Ingling, Richard Murray, Christopher Broussard, et al. Whats new in psychtoolbox-3. *Perception*, 36(14):1, 2007.

Author Biography

Gennady Livitz received his master degree in Electrical Engineering from MADI University, Moscow, in 1979 and his Ph.D in Cognitive and Neural Systems from Boston University in 2010. Since then he worked on modeling vision and color perception at Neuromorphics Lab at Boston University and at Computational Vision Lab at Northeastern University. His current work at iRobot Corporation is focused on VSLAM for visually guided automated robots.