Learning to Find Occlusion Regions

Ahmad Humayun Oisin Mac Aodha Gabriel J. Brostow
University College London
http://visual.cs.ucl.ac.uk/pubs/learningOcclusion/

Abstract

For two consecutive frames in a video, we identify which pixels in the first frame become occluded in the second. Such general-purpose detection of occlusion regions is difficult and important because one-to-one correspondence of imaged scene points is needed for many tracking, video segmentation, and reconstruction algorithms. Our hypothesis is that an effective trained occlusion detector can be generated on the basis of i) a broad spectrum of visual features, and ii) representative but synthetic training sequences. By using a Random Forest based framework for feature selection and training, we found that the proposed feature set was sufficient to frequently assign a high probability of occlusion to just the pixels that were indeed becoming occluded. Our extensive experiments on many sequences support this finding, and while accuracy is certainly still scene-dependent, the proposed classifier could be a useful preprocessing step to exploit temporal information in video.

1. Introduction

Different parts of a scene become occluded and disoccluded over the course of a video, confounding attempts to compute the motion field. Motions that are fast with respect to a camera's frame rate can cause large *regions* of pixels to temporarily disappear from view, while slower motions hide only the pixels on an object's leading occlusion *boundary*. We focus on the former because 3D occlusions are so prevalent in video. Though occlusion regions are occasionally ignored, some applications attempt to cope with them by treating them as outliers. For example, when estimating optical flow in a neighborhood, it is typical for an energy function to balance the model's desire for uniform flow against the evidence that the neighborhood's flow is discontinuous. The challenge is increased when ambiguous motion calls for substantial smoothing, yet occlusion can trivially excuse any mismatch between the data and the model.

Some algorithms posit that occlusions are correlated with intensity discontinuities. Though impractical for scenes with substantial texture, they either regularize less when the intensity changes, or not at all when dealing, for example, with a superpixel boundary [32]. Our experiments confirm that image gradients are a useful cue, but that many other cues, suggested individually and in groups by previous researchers, are also very revealing and correlated with occluded pixels. Our main contribution is that supervised learning with a broad feature vector, formed from a spectrum of cues, allows our algorithm to compute a probabilistic per-pixel occlusion map for each pair of images in a sequence.

2. Related Work

The detection of occlusion *regions* for their own sake has received somewhat less attention than occlusion in the context of general motion or depth segmentation, layer extraction, and identification of occlusion *boundaries*. Here we review only the most directly related research, where occlusion was computed either explicitly, or implicitly as part of other objectives.

Occlusion Boundaries Occlusion *boundaries* are distinct from occlusion regions, and are also very useful because they can hint at ordered depth, motion direction, and scene context. Given multiple frames, one approach, proposed by Fleet *et al*. [12], created a generative non-linear model for motion boundaries. Their Bayesian formulation explained local image motion in terms of multiple competing nonlinear models of, among others, translational motion and motion discontinuities. They identified occlusion using the differences between background and foreground velocity, normal to the edge.

T-junctions are known to be important indicators for occlusion boundaries. Apostoloff and Fitzgibbon [2] learned the appearance of T-junctions using SIFT features [26] on spatiotemporal patches with a Relevance Vector Machine. They find occlusion boundaries by identifying edges close to T-junctions on a spatiotemporal slice. Stein and Hebert [37] showed that these boundaries can be learned directly using AdaBoost with appearance and motion cues from multiple frames, typically 8-20 frames. These motion features depend on the translational motion of intensity-based superpixels, and the speed of those boundary fragments as measured by spatiotemporal filters. Using loopy belief propagation, their global boundary model converges to a consensus on the probable labeling of all boundaries. To interpret a scene, He and Yuille [14] demonstrated that a global boundary model is not essential, especially when dealing primarily with egomotion. They achieved comparable results to [37] by combining *pseudo-depth* and a local edge map, also from intensity superpixels, in a multilayered perceptron. Sundberg *et al.* [40] improve over these techniques by comparing motion differences each side of static boundaries predicted using *motion gradients*. A requirement of these methods is the use of more than two frames to identify occlusion boundaries in an image pair. Although our feature vector borrows from these techniques, it: (1) is computed from only two frames in a video; (2) makes no assumptions about the type of motion being observed; (3) completely avoids committing to intensity-based superpixel boundaries; and (4) combines multiple motion and intensity-based cues.

Methods have also been proposed to find occlusion boundaries in single frames. With no motion information, occlusion boundary detection in a single image remains an inherently ambiguous problem. Hoiem *et al.* [16] used an initial over-segmentation to enumerate possible object boundaries when performing automatic single-view reconstruction. Saxena *et al.* [34] also used a supervised learning approach to classify boundaries between superpixels but relaxed some of the strong planar assumptions of the former technique to cope with more general scenes. Using a global boundary model similar to [37], Hoiem *et al.* [17] used boundary, region, and surface layout cues for classification. Assuming a camera height and focal length, they use geometric class labels to get depth cues.

Layers While we make no assumptions about the number of independent motions in a scene, doing so helps organize a video into layers. Layers of video with occlusions was pioneered by Wang and Adelson [42]. One approach is to model each input image as a layered composition of a fixed number of flexible sprites [21]. Variational EM was used to infer the sprite's state, including the obstructed pixels. To make this technique more robust to occlusion, Frey *et al.* [13] extended it by considering the variability of the layer's motion in each sprite as a manifold. Kumar *et al.* [24] also explored an unsupervised approach with a generative layered representation for motion segmentation. Their latent image representation, which is learned over independent frames, also explicitly models occlusions. Our probability of occlusion is independent perpixel, as we have no expectation of contiguous regions or a pre-determined family of allowable motions.

Xiao and Shah [46] compute affine and projective transformations of planar regions while detecting occlusion pixels, before segmenting the scene into motion layers. They pose multi-frame motion segmentation as an energy minimization problem which has penalties for occlusion and occlusion order constraints on each consecutive frame pair.

Ayvaci and Soatto[4] segment motion using a superpixel graph, where a min-cut reveals intensity-based segments which can be clustered together. Part of the unary term in their graph penalized the photoconsistency residual, which helps identify partially or fully occluded superpixels. Their method alternates between motion segmentation and occlusion detection until the energy converges.

Stereo While we cope with both egomotion and moving objects, the stereo research community has attempted to explicitly find occlusion *regions* for rigid scenes [22]. Yang *et al.* [47] alternate between estimating disparity in one view based on the occlusion in the other, and estimating occlusion based on disparity. During optimization, if the disparity of a pixel in the right and left images is not consistent, the pixel is labeled as occluded. Sun *et al.* [39] uses occlusion information to update their data term toward the end of their optimization. Due to the difficulty of estimating disparity in textureless regions, both papers assume regions with similar color have similar disparity, which is consistent with [6]. In this way, the assumption is made that occlusion regions coincide with the boundaries of these color regions. To regularize in regions of low texture, a plane fitting step is performed, which can result in inaccuracies when dealing with non-rigid and non-planar scenes.

Occlusion-Aware Optical Flow Occluded pixels have undefined flow, so good estimates of flow depend on occlusion and vice versa. Heitz and Bouthemy [15] addressed the dense motion estimation problem in a multimodal formulation. Using complementary constraints, their method estimates optical flow while preserving motion discontinuities and presumed occlusions. The relationship between observation fields and motion labels is determined using an MRF, where local motion constraints are ignored in regions where occlusions are likely to occur.

Most current flow techniques use a regularization step to propagate flow into regions of low texture. On their own, such techniques arbitrarily diffuse flow to regions of occlusion, even though flow is essentially not applicable to such regions. Motion estimation techniques explicitly dealing with this situation insert an occlusion penalty term into the energy. Alvarez *et al.* [1] proposed one such solution, computing symmetrically dense optical flow which is consistent from I_1 to I_2 and I_2 to I_1. This is achieved by minimizing an energy function which respects occlusion. They initialize their minimization with standard one-way flow in each direction. Areas with inconsistent flow are labeled as occluded. They achieved positive results on synthetic sequences, but do not attempt to deal with large displacements.

The energy functional proposed by Ince and Konrad [19] includes three terms: photometric consistency weighted by inverse disparity in reverse and forward flow, a shrinking term to avoid degenerate solutions where everything is occluded, and an anisotropic diffusion term to extrapolate flow into occlusion regions. The diffusion process is still driven by image gradients, so their method is severely hindered by highly textured areas. In comparison, Xiao *et al.* [45] addresses the shortcomings of the diffusion filter by dividing the variational updating model into two steps: the first produces intermediate flow by minimizing the data energy, and the second diffuses this intermediate flow. For occlusions, they add a heavyside function of the image residue to the energy functional in the first term. They use a multi-cue bilateral filter instead of the traditional anisotropic filter to avoid diffusing flow across occlusion boundaries.

Ayvaci *et al.* [3] treat residuals for occlusion and non-occlusion areas separately. Their energy function minimizes the area of occlusion while minimizing the negative log-likelihood of the non-occlusion residual. This becomes a convex problem by using an adaptively weighted ℓ_1 norm of their energy function, based largely on flow photoconsistency. However, these flow-based techniques still require parameter tuning as optimal values can be very scene specific. The success of these techniques motivates us to utilize optical flow computation as one of the significant components of our larger feature vector.

Occlusion Regions Kolmogorov and Zabih [23] incorporate Marr-Poggio's [28] uniqueness principle into their graph-cut algorithm, stipulating that each pixel in one frame shall correspond to at most one pixel in the other frame. A pixel with no match is then considered as occluded. Jodoin *et al.* [20] built on this principle to make it more robust to noise. Using a method similar to Ince and Konrad [18], they generated a coarse occlusion map where a pixel is marked occluded if fewer than two pixels' intensities in its neighborhood are within a certain Euclidean distance. They iteratively improve their occlusion map by adjusting a color segmentation map. Lobaton *et al.* [25] also employed color information in finding occlusions. They introduced an image homeomorphism criterion for detecting local occlusions when objects and backgrounds can clearly be segmented into connected components based on distinct color distributions. The technique was demonstrated on real sequences and unusually, is designed to even cope with deformations. We too make no assumptions about the types of deformation in our videos, but are not restricted to scenes where foreground segmentation is obvious.

3. The Occlusion Classification Algorithm

We formulate occlusion region detection as a supervised binary classification problem. Our classifier for each data point, represented as a feature vector f, is defined as

$$\mathcal{O} := \left\{ (f_i \to L_i) \mid f_i \in \mathbb{R}^d,\, L_i \in [0,1] \right\}_{i=1}^{t}, \qquad (1)$$

where label $L = 1$ indicates a region of occlusion, d is the length of the feature vector, and t is the number of training samples available. We use Random Forests [8] for classification because they inherently perform feature selection, and our feature vectors are quite long by design. Given high quality labeled occlusion training data, the core of this algorithm hinges on filling f with cues that *may* correlate well with occlusion regions. We now present the sets of features that f comprises, and explanations of how these features were chosen. This feature set is not definitive, but has been chosen to cover a broad spectrum of cues.

Overall, flow plays a key role in indicating occlusion regions, so it is used throughout our feature vector. Even though we identify motion cues as necessary, no single algorithm is best at indicating occlusions. Like Mac Aodha *et al.* [27], we use a set $a \in [1, k]$ of flow algorithms to find per-pixel flow confidence based on supervised classification. This helps capitalize on situations where the flow algorithms are in consensus, or identify regions where the motion fields seem arbitrary. We employ a different set of $k = 4$ algorithms. [43, 44, 38, 10] are chosen for their high ranking on a spectrum of test data, with implementations provided by their respective authors.

Occlusion Boundaries Many occlusion boundaries lie close to object edges. A distance transform on an edge detector's output indicates to the forest the proximity of such a boundary. We use the Canny edge detector [11] on I_1, the first frame of the two frame sequence being considered, with the hysteresis thresholds τ_{ED} provided by `MATLAB`, so

$$f_{\text{ED}}(\mathbf{x}, z) = \text{distTrans}\left(\|\nabla I_1(z)\| > \tau_{\text{ED}} \right), \qquad (2)$$

where $f_{\text{ED}}(\mathbf{x}, z)$ gives the feature value at pixel $\mathbf{x}$ at scale-space level z. Further, some methods [17, 37, 14] instead use the P_{b} edge map [29] as an initial guess for finding occlusion boundaries. Computing the edge distance after thresholding this map with τ_{PB} gives the feature

$$f_{\text{PB}}(\mathbf{x}, z) = \text{distTrans}\left(P_{\text{b}}\left[I_1(z) \right] > \tau_{\text{PB}} \right), \qquad (3)$$

which strongly correlates with other object boundaries. High values are assigned to both the occluding and occluded surface, which is a typical quality of weak learners.

Photo and Texture Consistency Most flow algorithms work on the brightness constancy assumption, so cross-checking constancy in a feature can help reveal occlusions. By advecting the pixels in I_1 by u_a, the result from flow algorithm a, we compute the consistency in value with the corresponding pixel in I_2. This is similar to the image residual, *i.e.* photo consistency term used in the energy functional of some flow and occlusion detection methods [3, 45, 19]:

$$f_{\text{PC},a}\left(\mathbf{x}, z\right) = \left| I_1\left(\mathbf{x}, z\right) - \text{bicubic}\left(I_2\left(\mathbf{x} + u_a(\mathbf{x}, z), z\right)\right)\right|. \tag{4}$$

Note that a high value is assigned to pixels where $\mathbf{x} + u_a(\mathbf{x}, z)$ is out of bounds of the frame. This allows the forest to notice occlusions due to changes in field of view (FOV).

Similarly, inconsistency in texture can also reveal occlusions. Brox [9] proposed a condensed texture filter suited for discrimination tasks. These textures are computed over both I_1 and I_2. From texture patches around each pixel, [9] proposes to compute texture difference as

$$f^n_{\text{ST},a}\left(\mathbf{x}\right) = \tfrac{1}{D}\textstyle\sum_{d=1}^{D}\left(\frac{\mu_n(\mathbf{T}_{1,d}(\mathbf{x})) - \text{bicubic}(\mu_n(\gamma^a_d(\mathbf{x})))}{\sigma_n(\mathbf{T}_{1,d}(\mathbf{x})) - \text{bicubic}\left(\sigma_n\left(\gamma^a_d(\mathbf{x})\right)\right)}\right)^2, \tag{5}$$

where $\mathbf{T}_{1,d}\left(\mathbf{x}\right)$ and $\mathbf{T}_{2,d}\left(\mathbf{x}\right)$ denote the d^{th} texture filter response at pixel $\mathbf{x}$ for I_1 and I_2 respectively, and $\gamma^a_d(\mathbf{x}) \equiv \mathbf{T}_{2,d}\left(\mathbf{x} + u_a(\mathbf{x})\right)$ gives texture values in I_2 for each corresponding pixel $\mathbf{x}$ in I_1 using flow. The texture feature depth is $D = 5$. $\mu_n(\cdot)$ and $\sigma_n(\cdot)$ denote the mean and standard deviation of the texture in an $n \times n$ neighborhood. Using the advected texture, we also compute the Mahalanobis distance per pixel between the two texture features,

$$f_{\text{STm},a}\left(\mathbf{x}\right) = \sqrt{\sum_{d=1}^{D}\left[\mathbf{T}_{1,d}\left(\mathbf{x}\right) - \text{bicubic}\left(\gamma^a_d(\mathbf{x})\right)\right]^2 / \sigma^2_d}, \tag{6}$$

where σ^2_d is the variance of filter response d over both $\mathbf{T}_{1,d}$ and $\mathbf{T}_{2,d}$.

Flow Features We assume that many depth discontinuities that are likely to create occlusions are also flow discontinuities. The gradient of the median flow of all the flow algorithms is one simple measure of such discontinuities:

$$f_{\text{TG},x}\left(\mathbf{x}, z\right) = \left\|\nabla \bar{u}_x\right\|, \quad f_{\text{TG},y}\left(\mathbf{x}, z\right) = \|\nabla \bar{u}_y\|. \tag{7}$$

Since temporal gradient is computed only over two pixels, it is unable to detect proximal flow changes. To address this concern, we examine the spatial variance of flow angles $\theta_a(\mathbf{x}, z) = \arctan\left[u_{x,a}(\mathbf{x}, z)/u_{y,a}(\mathbf{x}, z)\right]$, within a given window size. Here, the change in direction of flow within a conservative window hints at different surfaces around a pixel. For each flow algorithm, we compute an angle variance over scale-space,

$$f^n_{\text{AV},a}\left(\mathbf{x}, z\right) = \mathbf{E}\left[\left(\theta_a(\mathbf{x}_i, z) - \mathbf{E}\left[\theta_a(\mathbf{x}, z)\right]\right)^2\right], \tag{8}$$

where $\mathbf{x}_i \in \mathcal{N}_{\text{AV}}$ samples an $n \times n$ square neighborhood around $\mathbf{x}$. Similarly, with the flow norm $M_a(\mathbf{x}, z) = \|u_a(\mathbf{x}, z)\|$, we compute the spatial variance of flow length around a pixel

$$f^n_{\text{LV},a}\left(\mathbf{x}, z\right) = \mathbf{E}\left[\left(M_a(\mathbf{x}_i, z) - \mathbf{E}\left[M_a(\mathbf{x}, z)\right]\right)^2\right], \tag{9}$$

where $\mathbf{x}_i \in \mathcal{N}_{\text{LV}}$ is the window centered at $\mathbf{x}$. Note that both the angle and length variance features will also fire at disocclusions. We also compute the time it would take for pixels on diagonally opposite sides of a neighborhood to arrive on the center pixel. This is computed by projecting the flow of each pair of pixels onto the line connecting them. With an $n \times n$ window, time to collision is computed for all $(n^2 - 1)/2$ pixel pairs. Given a center pixel $\mathbf{x}$, the two diagonally opposite pixels are given by $\mathbf{x} \pm \mathbf{r}$, and the vector between them is $\vec{v} = 2(\mathbf{x} - \mathbf{r})$, so

$$\Psi_a(\mathbf{x}, \mathbf{r}, z) = \tfrac{2\|\mathbf{r}\|}{\text{proj}_{\vec{v}}(u_a(\mathbf{x}+\mathbf{r}, z)) + \text{proj}_{\vec{v}}(u_a(\mathbf{x}-\mathbf{r}, z))}, \tag{10}$$

$$f^n_{\text{CS},a}\left(\mathbf{x}, z\right) = \max\left\{\ \Psi_a(\mathbf{x}, \mathbf{r}_i, z) :\ \mathbf{r}_i \in \mathcal{N}_{\text{CS}}\right\} \tag{11}$$

where $\mathcal{N}_{\text{CS}}$ is half the $n \times n$ neighborhood, excluding $\mathbf{x}$. Since flow fields are invalid at regions of occlusion, flow computed here is untrustworthy. [27] demonstrated a principled metric of uncertainty. They trained a classifier where flow under a certain end point error (EPE) is labeled as *reliable* or not. The resulting posterior indicates flow confidence, which tends to be low at motion discontinuities and occluded pixels. We use their implementation to train $4 \times k$ (the number of flow algorithms) classifiers: two for $f_{\text{FC}_{\text{EPE}},a}$ based on EPE, and two $f_{\text{FC}_{\text{AE}},a}$ based on angular error, where the respective thresholds τ_{EPE} and τ_{AE} have high and low versions.

Flow confidence is a useful indicator since flow at occlusions varies arbitrarily from the ground-truth. Although some flow methods find occlusions and smooth flow in these areas, others produce fields by turning a blind eye to these regions. This indicates that across flow algorithms, results over occlusions can be quite arbitrary. A feature which indicates this variance in motion vectors across the k flow techniques is

$$f_{\text{FA}}\left(\mathbf{x}, z\right) = \text{E}\left[\left(\theta_{a_i}(\mathbf{x}, z) - \text{E}\left[\theta_a(\mathbf{x}, z)\right]\right)^2\right], \tag{12}$$

where $a_i \in [1, k]$. Similarly, the variance of motion vector length across flow algorithms is

$$f_{\text{FN}}\left(\mathbf{x}, z\right) = \text{E}\left[\left(M_{a_i}(\mathbf{x}, z) - \text{E}\left[M_{a_i}(\mathbf{x}, z)\right]\right)^2\right]. \tag{13}$$

Finally, to compensate for the noise in the temporal gradient feature, we cluster the flow vectors using intensity based superpixels. First, superpixels for I_1 are created using [30]. Average median flow is assigned to each superpixel, and gradient magnitude is taken over this new image to get coarse flow discontinuities. The discontinuities are dilated by a small gaussian kernel. This gives the flow-discontinuity feature f_{SP}.

Reverse Flow Features Flow in both directions between I_1 and I_2 can be computed. Given perfect bidirectional flow fields, a pixel advected from I_1 to I_2 using $u_a(\mathbf{x})$ and back using $u'_a(\mathbf{x}')$ should return the pixel to its original location. This is not true for occluded pixels, as they *flow* to undefined locations. Similar to the approaches in [1, 19, 31], we find this disparity in forward and reverse flow by taking the Euclidean distance between the original location and its position after reverse flow $\mathbf{x}'_a = \text{round}\left(\mathbf{x} + u_a(\mathbf{x}, z)\right)$,

$$f_{\text{RC},a}\left(\mathbf{x}, z\right) = \left\|\mathbf{x} - \left(\mathbf{x}'_a + u'_a(\mathbf{x}'_a, z)\right)\right\|. \tag{14}$$

$u'_a(\mathbf{x}'_a, z)$ is the reverse flow at scale-space level z using flow method a. Following the same argument, the reverse flow direction at non-occluded pixels should be nearly opposite of the forward flow direction, except in cases of errant flow in regions of occlusion. We compute this reverse flow angle consistency as

$$f_{\text{RA},a}\left(\mathbf{x}, z\right) = \left|\pi - \arccos\left[u_a(\mathbf{x}, z)\ \cdot\ u'_a(\mathbf{x}'_a, z)\right]\right|. \tag{15}$$

In all, we use a $d = 227$ dimensional feature vector f_i, which is computed for each pixel, using the flow algorithms set $K = \{1...4\}$, two scale-space choices $S_1 = \{1...4\}$ and $S_2 = \{1...10\}$, and parameters given in Section 4:

$$\begin{aligned} f_i = \{ & f_{\text{ED}}\left(\mathbf{x}, S_2\right),\ f_{\text{PB}}\left(\mathbf{x}, S_1\right),\ f_{\text{PC},K}\left(\mathbf{x}, S_1\right),\ f^n_{\text{ST},K}\left(\mathbf{x}\right), \\ & f_{\text{STm},K}\left(\mathbf{x}\right),\ f_{\text{TG}}\left(\mathbf{x}, S_2\right),\ f^n_{\text{AV},K}\left(\mathbf{x}, S_1\right),\ f^n_{\text{LV},K}\left(\mathbf{x}, S_1\right), \\ & f^n_{\text{CS},K}\left(\mathbf{x}, S_1\right),\ f_{\text{RC},K}\left(\mathbf{x}, S_2\right),\ f_{\text{RA},K}\left(\mathbf{x}, S_2\right),\ f_{\text{SP}}\left(\mathbf{x}\right), \\ & f_{\text{FA}}\left(\mathbf{x}, S_2\right),\ f_{\text{FN}}\left(\mathbf{x}, S_2\right),\ f_{\text{FC}_{\text{EPE}},K}\left(\mathbf{x}\right),\ f_{\text{FC}_{\text{AE}},K}\left(\mathbf{x}\right) \} \end{aligned} \tag{16}$$

4. Experiments

We investigated several different parameter settings for our features by training classifiers using only these features and observing the results. We used the dataset from [27] in addition to new sequences[1] during this optimization. After experimenting with a range of window sizes (up to $n = 9$) for $\{f^n_{\text{AV},a}\left(\mathbf{x}, z\right), f^n_{\text{LV},a}\left(\mathbf{x}, z\right), f^n_{\text{CS},a}\left(\mathbf{x}, z\right)\}$ we choose $n = 3$. For $f^n_{\text{CS},a}\left(\mathbf{x}, z\right)$ we experimented with min., max. and the variance in collision time, and found them to be equivalent to each other, and hence only used max.. We also did a similar experiment for $f^n_{\text{ST},a}\left(\mathbf{x}\right)$ with window sizes up to $n = 45$, and found that $n = 3$ gave the best ROC scores. We use values of 0.1 and 0.4 for τ_{PB} to exploit low and high confidence edges. To evaluate the flow confidence features, two thresholds were chosen for the angular error and EPE versions. The values of $\tau_{\text{EPE}} = \{1, 50\}$ and $\tau_{\text{AE}} = \{1°, 60°\}$ were chosen to measure confidence on a fine and coarse level. We choose the following parameters for the Random Forest: 35 maximum depth of a tree; 11 random features per node; and 105 trees. The resulting feature importance and computational costs are given in Figure 1.

[1] All training sequences were generated using the code from [27]. Our training data is available from our project webpage.

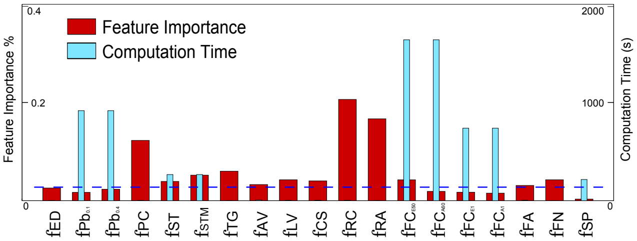


Figure 1. Feature importance returned by Random Forests and respective feature computational cost. The left scale gives the mean feature importance during leave-one-out tests; while the right scale gives the mean feature computation times. These values help us build our lean classifier, where we keep only features whose computation times are below the blue threshold.

4.1. Quantitative Results

We evaluate our algorithm on data where ground truth occlusions were available, reporting the area under the Receiver Operator Characteristics (ROC) curves and displaying the Precision-Recall (PR) curves. We detect occlusion regions caused by the camera changing its FOV seemingly easily. In addition to scores for Full Ground Truth (FGT), we also compare against Cropped Ground Truth (CGT), where those pixels that leave the frame are not counted.

Leave-One-Out Table 1 gives leave-one-out scores for the ground truth dataset. In general, we train on 13 sequences, but some are the same scene from two different viewpoints. Obviously, the leave-one-out experiments actually leave-two-out to test correctly in such situations. For all subsequent experiments unless otherwise stated, the classifier was trained on this synthetic dataset, because there were usually visible mistakes in the ground truth occlusion regions of other relevant datasets.

Figure 2 gives quantitative results for a subset of our dataset with PR curves. Our posteriors correlate well with the ground truth occlusion regions. We are not mislead by

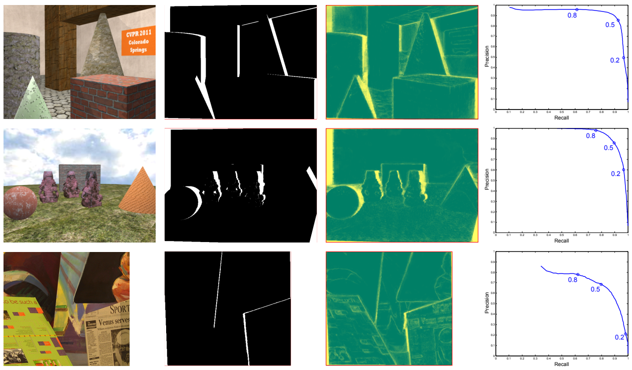


Figure 2. Quantitative results. Each row represents a different test example (BrickBox1t1, Mayan10, and Venus respectively) with the corresponding first image from the pair, ground truth occlusion, our probability of occlusion, and a PR curve for the FGT.

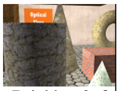
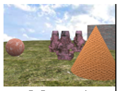
	 Crates1	 Crates2	 Robot	 Sponza1	 Sponza2	 Crates1txtr	 Brickbox1t1	 Brickbox2of	 Mayan1	 Text1
Single FGT	0.564	0.658	0.536	0.633	0.640	0.511	0.766	0.604	0.508	0.626
Lean CGT	0.826	0.758	0.851	0.945	0.814	0.981	0.979	0.641	0.952	0.973
Lean FGT	**0.950**	0.961	0.922	0.940	**0.929**	0.996	0.986	0.981	0.976	0.985
Our CGT	0.744	0.682	0.876	**0.947**	0.812	0.991	0.989	0.618	0.973	0.982
Our FGT	0.942	**0.969**	**0.936**	**0.947**	0.928	**0.997**	**0.992**	**0.991**	**0.986**	**0.991**

Table 1. Leave-one-out scores reported as area under ROC curve. The **first row** shows results when using the single image features; **second** and **third rows** gives results of our lean classifier on CGT and FGT respectively; **fourth** and **fifth rows** gives results for the full set of features on CGT and FGT.

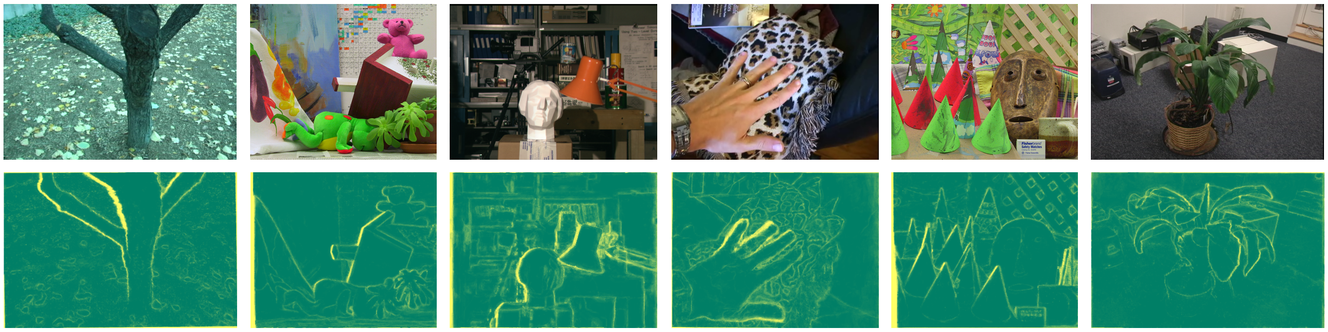


Figure 3. Qualitative results of our algorithm with data from [37, 5, 33, 36, 35]. Our posterior probability of occlusion is shown below the corresponding first frame from the image pair. For further results, please see the supplementary material.

scene texture, and while there is a small probability associated with some of the image gradients which do not get occluded, there is a much higher probability assigned to the actual occlusions. The advantage of our posterior is that it can be thresholded at any operating point, depending on the confidence required. We perform slightly worse on the Venus sequence, but as this ground truth was created by hand it may contain errors which decrease our apparent performance.

Additional qualitative results on natural sequences which have no ground truth are presented in Figure 3. For more results, please see the supplementary material.

Comparisons to Other Methods To emphasize the importance of using temporal information when classifying occlusion regions, we compare our results against a classifier trained only on features extracted from a single image. For this feature set, we ignored all the features which have a temporal element, training on the remaining ones. The first row in Table 1 reports scores for this single image occlusion detector. As expected, the results are better than chance but much worse than using temporal information.

We perform a quantitative analysis between our algorithm and that of Kolmogorov and Zabih [23]. Their algorithm is specially designed to detect occlusion in stereo image pairs, so was not intended for general scenes. Nevertheless, like [3] we use it as a baseline algorithm. For all sequences, the maximum number of iterations for their algorithm was set to 50, the min and and max disparity range for x and y were both set to -15 and 15, and λ was chosen automatically by their algorithm. There is presently no publicly available dataset specifically designed for occlusion regions in natural scenes with accurate ground truth. We use sequences from the Middlebury optical flow and stereo datasets [5], with hand relabeling of incorrect regions for some of the sequences, and removal of a 10 pixel border region. Additionally, we report results for our own synthetically generated sequences with trusted occlusion regions based on code from [27]. This data is made available on our website for future comparisons. Table 2 gives the precision and recall results for the two algorithms. [23] returns a binary occlusion mask, so to make a fairer comparison, we first compute their recall and precision, and report our precision at the closest equivalent recall rate. We outperform

	Venus	RubberWhale	Tsukuba	Mayan10	Crates1deg3h	Textl	BrickBox1t1
Recall [23]	0.60	0.23	0.44	0.36	0.15	0.82	0.51
Recall (Ours)	0.59	0.23	0.43	0.49	0.83	0.82	0.51
Precision [23]	0.63	0.31	0.58	0.33	0.10	0.68	0.49
Precision (Ours)	**0.69**	**0.47**	**0.85**	**0.99**	**1.0**	**0.88**	**0.96**

Table 2. Occlusion region labeling comparison between our algorithm and Kolmogorov and Zabih [23].

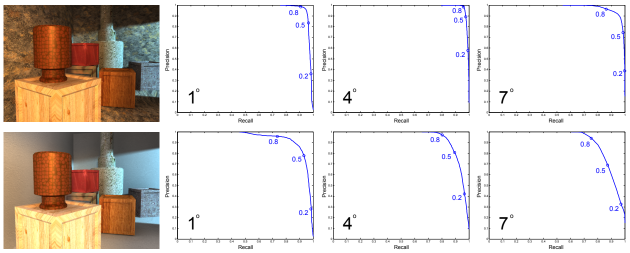


Figure 4. Results for varying scene texture and occlusion region size. The two rows represent the same scene with different background texture. For the second frame, the camera was rotated about the nearest corner of the wooden box in the foreground along the x axis for 1°, 4° and 7°. The PR in each column reports results on the FGT for different amounts of occlusion.

the baseline in all tested sequences.

Changes in Scene Texture and Occlusion Region Size In the final quantitative experiment, we tested how our algorithm's accuracy varies with scene texture and size of occlusion regions. We have modified a synthetic scene by making two texture variants and three occlusion size variants. The occlusions in the Crates1 sequence are due to egomotion. The PR curves in Figure 4 show our algorithm coping well with different amounts of occlusion. As the angle increases the performance decreases, but not dramatically. Lack of scene texture is a more noticeable challenge, due to the diminished performance of features based on flow.

5. Applications

We demonstrate a possible use for our occlusion region detection algorithm. We also present a sped up version of our classifier, which gives comparable results but at a significantly reduced computational cost.

Occlusion-Aware Over-Segmentation Many video processing pipelines use an initial over-segmentation as a preprocessing step [41]. These techniques typically group similar colored pixels while respecting image boundaries in a single frame. By ignoring occlusions in the scene, the superpixels may contain a mixture of foreground and background objects, or pixels that will disappear. These *mixed* superpixels can prove difficult to match between frames for tracking or segmentation. We have applied our occlusion posterior to create occlusion-aware superpixels.

The publicly available superpixel code from Mori *et al.* [30] uses the P_b boundary detector of [29] to provide an edge map. This detector attempts to classify boundaries in natural images by training on hand labeled edge maps. We augment this boundary map by combining it with our occlusion posterior. [30] expects boundaries and not regions, so we first convert our posterior into a boundary edge map. To regularize our posterior and create such a binary occlusion map, we pose and solve this as a graph labeling problem [7]. The occlusion posterior is used as the unary term, and the pairwise term is set at a constant weight of 0.25 for all experiments. Edges of this binary mask are then extracted and combined with the P_b boundary map by taking the maximum in each location. Figure 5 shows the results for an image pair of a natural scene with static camera and a moving foreground object. It can be seen that the introduction of our occlusion term produces an over-segmentation which respects image edges, region color, and occlusions.

Lean Feature Set Our current implementation achieves accurate results but at the expense of long computation times. Thanks to the broad spectrum of features examined and importance-assessed by our Random Forest, we can experiment with a stripped down version of our feature set. We removed families of features by examining the feature importance graph in Figure 1 along with the computational cost associated with computing each family. We keep the features whose combined computation time is under 2 mins, illustrated by the dashed horizontal line in the figure. Table 1 reports results for this *Lean* version of our algorithm for both full and cropped ground truth, and visual comparisons are in the supplementary material. With this reduced feature set, training time (excluding feature extraction) was reduced from 54 to 25 mins for approximately 80,000 datapoints (13 sequences). Then, feature extraction and testing takes 1.3 as compared to 124 mins for an image pair.

6. Conclusions

We have presented a general purpose method of detecting occlusion regions in image sequences. We do not make assumptions about the scene type and have illustrated results for occlusion due to egomotion and both rigid and non-rigid scene motion in real sequences. By posing the problem in a supervised learning context, we learn features from the data which correlate with occlusion. This gives us an advantage over existing methods which rely on manual tuning of parameters.

We have evaluated our algorithm quantitatively on synthetic data and compared our performance to two baseline systems (single image and stereo based methods). We have also evaluated the impact of texture and motion extent on the accuracy of occlusion region detection. Due to the feature selection step in our algorithm, we are not limited to finding occlusions which only coexist with image edges. However, our performance does decrease with a reduction in scene texture.

A lightweight version of our algorithm which trains on a reduced set of the full feature vector is proposed. It gives performance comparable to our full version, but at a much reduced computational cost. Future work will include the development of a spatio-temporal superpixel algorithm which explicitly takes our occlusion posterior into account, and exploration of non-local features to help cope with drastic motions and very low texture.

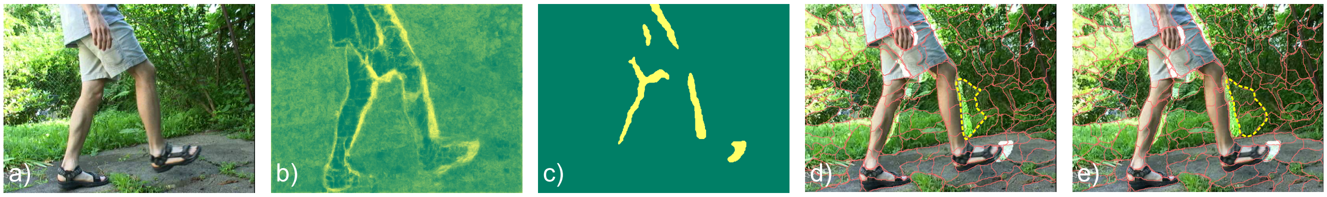


Figure 5. Occlusion-aware over-segmentation. a) First frame from a two frame sequence of a person walking forward [37]. b) The occlusion posterior of our algorithm has correctly labeled the region in front of the right and left legs as being occluded in the second frame. Note that the algorithm does not label the dis-occluded regions at the backs of the leg as occluded. c) Regularized posterior using graph cuts. d) Over-segmentation using only boundary information from a single frame. e) Over-segmentation using additional occlusion region information. By comparing the occluded regions (superimposed with higher intensity) to the right of each leg and hand in d) and e) it can be seen that the occlusion-aware over-segmentation of e) respects both the image and occlusion boundaries.

Funding for O. Mac Aodha was partly provided by the NUI Travelling Studentship in the Sciences.

References

[1] L. Alvarez, R. Deriche, T. Papadopoulo, and J. Sánchez. Symmetrical dense optical flow estimation with occlusions detection. *IJCV*, 75(3):371–385, 2007.

[2] N. Apostoloff and A. Fitzgibbon. Learning spatiotemporal T-Junctions for occlusion detection. In *CVPR*, 2005.

[3] A. Ayvaci, M. Raptis, and S. Soatto. Occlusion detection and motion estimation with convex optimization. In *NIPS*. 2010.

[4] A. Ayvaci and S. Soatto. Motion segmentation with occlusions on the superpixel graph. In *ICCV Workshop on Dynamical Vision*, pages 727–734, 2009.

[5] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. In *ICCV*, 2007.

[6] A. F. Bobick and S. S. Intille. Large occlusion stereo. *IJCV*, 33(3):181–200, 1999.

[7] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 26(9):1124–37, 2004.

[8] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[9] T. Brox. *From pixels to regions: partial differential equations in image analysis*. PhD thesis, Faculty of Mathematics and CS, Saarland University, 2005.

[10] T. Brox and J. Malik. Large Displacement Optical Flow: Descriptor Matching in Variational Motion Estimation. *PAMI*, 99, 2010.

[11] J. Canny. A computational approach to edge detection. *PAMI*, 8(6):679–698, 1986.

[12] D. Fleet, M. Black, and O. Nestares. Bayesian inference of visual motion boundaries. pages 139–173, 2003.

[13] B. J. Frey, N. Jojic, and A. Kannan. Learning appearance and transparency manifolds of occluded objects in layers. In *CVPR*, 2003.

[14] X. He and A. Yuille. Occlusion Boundary Detection Using Pseudo-depth. In *ECCV*, 2010.

[15] F. Heitz and P. Bouthemy. Multimodal estimation of discontinuous optical flow using Markov random fields. *PAMI*, 15(12):1217–1232, 1993.

[16] D. Hoiem, A. A. Efros, and M. Hebert. Automatic Photo Pop-up. In *ACM SIGGRAPH*, 2005.

[17] D. Hoiem, A. A. Efros, and M. Hebert. Recovering Occlusion Boundaries from an Image. *IJCV*, pages 1–19, 2010.

[18] S. Ince and J. Konrad. Geometry-based estimation of occlusions from video frame pairs. In *IEEE ICASSP*, volume 2, 2005.

[19] S. Ince and J. Konrad. Occlusion-aware optical flow estimation. *TIP*, 17(8):1443–1451, 2008.

[20] P. M. Jodoin, C. Rosenberger, and M. Mignotte. Detecting half-occlusion with a fast region-based fusion procedure. In *BMVC*, 2006.

[21] N. Jojic and B. J. Frey. Learning flexible sprites in video layers. In *CVPR*, 2001.

[22] S. B. Kang, R. Szeliski, and J. Chai. Handling occlusions in dense multi-view stereo. In *CVPR*, 2001.

[23] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *ICCV*, 2001.

[24] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Learning layered motion segmentations of video. *IJCV*, 76(3):301–319, 2008.

[25] E. Lobaton, R. Vasudevan, R. Bajcsy, and R. Alterovitz. Local occlusion detection under deformations using topological invariants. In *ECCV*. 2010.

[26] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[27] O. Mac Aodha, G. J. Brostow, and M. Pollefeys. Segmenting video into classes of algorithm-suitability. In *CVPR*, 2010.

[28] D. Marr and T. Poggio. Cooperative computation of stereo disparity. Technical report, 1976.

[29] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(5):530–549, 2004.

[30] G. Mori, X. Ren, A. A. Efros, and J. Malik. Recovering human body configurations: combining segmentation and recognition. In *CVPR*, 2004.

[31] A. S. Ogale, C. Fermuller, and Y. Aloimonos. Motion Segmentation Using Occlusions. *PAMI*, 27:988–992, 2005.

[32] X. Ren and J. Malik. Learning a classification model for segmentation. *ICCV*, 2003.

[33] P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. In *CVPR*, 2006.

[34] A. Saxena, M. Sun, and A. Y. Ng. Make3D: Learning 3D Scene Structure from a Single Still Image. *PAMI*, 31:824–840, 2009.

[35] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *CVPR*, 2003.

[36] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Stereo and Multi-Baseline Vision*, 2001.

[37] A. N. Stein and M. Hebert. Occlusion Boundaries from Motion: Low-Level Detection and Mid-Level Reasoning. *IJCV*, 82(3):325–357, 2009.

[38] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *CVPR*, 2010.

[39] J. Sun, Y. Li, S. B. Kang, and H. Shum. Symmetric stereo matching for occlusion handling. In *CVPR*, 2005.

[40] P. Sundberg, T. Brox, M. Maire, P. Arbeláez, and J. Malik. Occlusion Boundary Detection and Figure/Ground Assignment from Optical Flow. In *CVPR*, 2011.

[41] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr. Videotrace: rapid interactive scene modelling from video. *ACM Trans. Graph.*, 2007.

[42] J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *IEEE TIP*, 3(5):625–638, 1994.

[43] A. Wedel, T. Pock, C. Zach, H. Bischof, and D. Cremers. An improved algorithm for TV-L1 optical flow. *Statistical and Geometrical Approaches to Visual Motion Analysis*, pages 23–45, 2009.

[44] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L1 Optical Flow. In *BMVC*, 2009.

[45] J. Xiao, H. Cheng, H. Sawhney, C. Rao, and M. Isnardi. Bilateral filtering-based optical flow estimation with occlusion detection. In *ECCV*. 2006.

[46] J. Xiao and M. Shah. Motion layer extraction in the presence of occlusion using graph cuts. *PAMI*, 27(10):1644–1659, 2005.

[47] Q. Yang, L. Wang, R. Yang, H. Stewenius, and D. Nister. Stereo Matching with Color-Weighted Correlation, Hierarchical Belief Propagation, and Occlusion Handling. *PAMI*, 31:492–504, 2009.