A Learning Based Framework for Depth Ordering

Zhaoyin Jia[1], Andrew Gallagher[2], Yao-Jen Chang[3], Tsuhan Chen[1]
[1]School of Electrical and Computer Engineering, Cornell University
[2]Eastman Kodak Company, [3]Siemens Corporate Research
zj32@cornell.edu, andrew.c.gallagher@gmail.com, yao-jen.chang@siemens.com, tsuhan@ece.cornell.edu

Abstract

Depth ordering is instrumental for understanding the 3D geometry of an image. We as humans are surprisingly good at depth ordering even with abstract 2D line drawings. In this paper we propose a learning based framework for discrete depth ordering inference.

Boundary and junction characteristics are important clues for this task, and we have developed new features based on these attributes. Although each feature individually can produce reasonable depth ordering results, they still have limitations, and we can achieve better performance by combining them. In practice, local depth ordering inferences can be contradictory. Therefore, we propose a Markov Random Field model with terms that are more global than previous work, and use graph optimization to encourage a globally consistent ordering. In addition, to produce better object segmentation for the task of depth ordering, we propose to explicitly enforce closed loops and long edges for the occlusion boundary detection.

We collect a new depth-order dataset for this problem, including more than a thousand human-labeled images with different daily objects in various configurations. The proposed algorithm gives promising performance over conventional methods on both synthetic and real scenes.

1. Introduction

Depth estimation can be instrumental to a variety of vision tasks, such as segmentation [1] [12], object recognition [6] [9], and scene understanding [5] [10] [16]. For some purposes, instead of estimating the exact depth value, it may suffice to derive the relative depth ordering of the objects in an image. Humans are adept at this task: in Fig. 1 (a), we may not exactly know how far these objects are, but we can understand the depth ordering of the objects: the mouse is on the top, and then comes the book, and the laptop is deeper in the pile supported by the table. The depth ordering not only gives us a coarse interpretation of the 3D geometry of the objects, but also enables us to interact further with the scene, e.g. we need to remove the mouse and the stapler in order to manipulate the book.

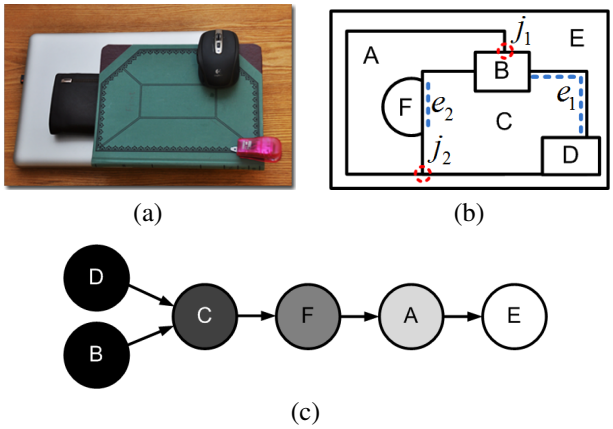
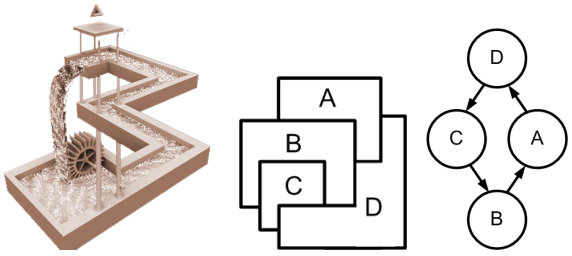


Figure 1. (a) Given one image, humans can infer the depth ordering of each object, and even with (b) very abstract line-drawing segments. Motivated by how humans reason about the depth ordering from junctions and boundaries, we develop an algorithm to do that. Our algorithm produces the depth ordering that represented in the form of a graph as in (c), where each node corresponds to one segment, and the directed edge means one segment is in front of another. The depth is colored in a way that the closer an object is, the darker it appears.

Humans have no trouble inferring the depth order even when the image is extremely abstract with only line drawings [3], such as Fig. 1 (b). We still understand that segment B is in front of segment A and C, segment D is in front of segment C, C is in front of F and so on. If we use "$\rightarrow$" to indicate the "in front of" relation, then we have $D \rightarrow C$; $B \rightarrow C \rightarrow F \rightarrow A \rightarrow E$. Early works from Barrow et al. [2] and Waltz et al. [18] present rule-based algorithms to understand 3D geometry in abstract images.

These examples inspire us to investigate the features that determine how we perceive the image depth ordering. Line drawings take out all the color, texture, and semantic high-level interpretation of the image. Clearly in this situation, only two types of information are available, i.e., boundaries and junctions, such as e_1, e_2, j_1, j_2 in Fig. 1 (b). However, depth ordering based on this information is not easily captured by hand-crafted rules, particularly in complex scenar-



(a) (b) (c)

Figure 2. (a) The Escher Waterfall shows that local reasoning cannot ensure the global consistency. (b) The same is true for depth ordering: although we can determine the pairwise relation between any two segments, it is difficult to decide the global depth order, and the corresponding depth order graph (c) forms a loop.

ios. Therefore we adopt a data-driven approach to handle its complexity. We design new features on boundaries and junctions, and use them as the basis to learn depth ordering.

Inferring the depth order from junction or boundary individually has some natural flaws, however. For example in Fig. 1 (b), junction j_1 and j_2 have the same T-shape, but imply inverse depth orders. Boundary e_2 is a straight line and provide little information by itself. Therefore we must combine these different features to form a better feature set.

Furthermore, having inferred local depth orders from the combined feature sets, we need to ensure the global consistency across the segments. Simply aggregating the local decisions can lead to an invalid understanding of the scene, and the famous Escher Waterfall in Fig. 2 (a) gives a vivid illustration for height perception. This point carries over to depth ordering, and Fig. 2 (b) gives one similar example: locally, we can easily determine the relative depth order between any two segments, such as $D \rightarrow C$, $C \rightarrow B$, $B \rightarrow A$ and $A \rightarrow D$. However, when aggregated, it is not a valid depth ordering, i.e. it forms a depth order graph with a loop, as shown in Fig. 2 (c). Therefore, to ensure global consistency in the depth ordering, we propose a Markov Random Field based algorithm to infer the most likely depth ordering and penalize the invalid order of segments. With this algorithm, global consistency is encouraged through message passing, which in turn enables better performance.

In addition, a reliable segmentation is an essential preparation for depth ordering. For natural images, we follow [7] to detect occlusion boundaries and generate object segments. We discover that, in many scenarios, the occlusion boundaries are not only locally continuous, but also form a closed loop to enclose the object. At the same time, the edges connected to and inside of this loop are less likely to be actual occlusion boundaries. We enforce this constraint, which is a more global enforcement than local continuity and leads to better object segmentation for depth ordering.

We collected a new depth order dataset with over a thousand images displaying different arrangements of various objects. Each image is manually segmented and includes depth information from Kinect. We tested different algorithms on this and two other datasets: one synthetic dataset and one with natural images [7]. Experiments proved the effectiveness of our proposed new features, and show that our proposed algorithm reliably outperforms the baselines.

To summarize, our major contributions are:

1. New features (on junctions and boundaries) and a learning based framework for the depth ordering task.
2. A novel approach to globally encourage the depth order consistency through a graphical model.
3. A new depth ordering dataset including more than 1000 images with human segmentation and depth information.
4. A new approach that favors closed loops for occlusion boundary detection.

2. Related work

Our work assumes that the scene is composed of objects in distinct depth order, and is closely related to the works from Dimiccoli et al. [4] and Palou et al. [13], which infer the depth ordering from an elaborate set of rules on T-junctions. Our work differs and improves upon previous works in the following aspects: a) in these works, the rules of inferences are designed without any learning process. They work in certain settings, but may not adapt to new environments. On the contrary, our approach is a learning-based framework and data-driven. b) Their algorithms focus only on the angles in T-junctions, while we show that combining boundary features with junctions is necessary and achieves better results. c) When aggregating local decisions to produce a global ordering, these works handle contradictions by dropping orders with the lowest predicted beliefs. We formulate this task as a graph inference problem, which achieves global consistency more accurately with the help of graphical model optimization.

Depth ordering is related to the boundary ownership or the figure and ground assignment problem [7] [14] [15] [17]. However we consider that they are non-trivially different tasks and produce different results. Figure and ground assignment is usually based on each edge as presented by Ren et al. [15], while depth ordering is based on segments. As a result, their work places more focus on features from edges, while we use a complementary feature set of junctions and boundaries. Depth ordering also introduces new problems, such as global consistency in depth, that may not exist for the figure/ground assignment problem.

Another approach is to infer depth based on high-level understanding of the scene, as in Hoiem et al. [7] and Liu et al. [10]. They parse an image into different semantic labels, such as “ground”, “sky”, etc, upon which they infer the depth mainly based on the connecting edge between the object and the ground plane. In their works, usually there is no need for encouraging the global consistency. The se-

mantic labels can largely solve this problem, like "ground" always supports "vertical surfaces", and they are placed before "sky". However, these geometric contexts may not always be applicable, such as shown in Fig. 1. In particular, these algorithms excel in natural scenes but fall short with micro objects or plan views, or may have difficulty in estimating the depth when "ground" falls outside of the image. Our algorithm complements this shortage well and aims to achieve reliable depth ordering from low level features without specific context.

Saxena et al. [16] propose to learn a regression for depth based on super-pixel features, and produce a continuous depth estimation. In contrast, our problem is based on occluded segments. The tasks and the approaches are significantly different. We believe we are able to achieve more meaningful depth relation between objects from reasoning the occlusions.

3. Local depth ordering

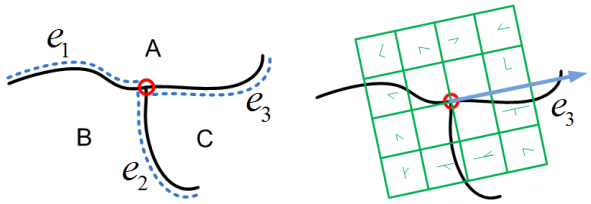
We first detect the occlusion boundaries in one image, and based on them we transform this image into segments. Then we compute features for depth ordering, build the depth order graph and assign a discrete depth value to each segment. We mainly rely on two sets of features for depth ordering: features on the T-junction (**pJF**) and on the boundary (**pBF**).

3.1. Junction feature

A T-junctions is where three boundaries and three segments meet, illustrated in Fig. 3 (a), and we aim to identify which segment is in front of the other two. Note that classifying which segment is in front is identical to classifying which one out of the three boundaries is behind, because the segments that are attached to this 'behind boundary' are also behind (see Fig. 3 (a)). In the following we will first classify this behind boundary, and then convert the result to the segment depth ordering.

Angle: A perfect T-junction will include one $180°$ angle between two boundaries, indicating the segment within is in front, and two $90°$ angles, indicating the segments are behind. We include these angles as our features. First, for each boundary e inside a junction, we fit a boundary vector $\vec{v}(e)$ to calculate its direction, shown in Fig. 3 (b), and calculate the angles from $\vec{v}(e)$ to the other two boundary vectors: $\theta_1, \theta_2 \in [0, \pi]$. We record them as a two-dimension feature $f_a(e)$ for boundary e within in this junction.

Texture: Junctions have different appearance in natural images, and thus using angles alone can be unreliable, so we also capture the texture information of a junction using an oriented SIFT descriptor [11]. SIFT descriptors can record the edge distributions within a junction, while tolerating some appearance variation by using histogram. The SIFT descriptor is centered at the junction, and aligned with every boundary vector $\vec{v}(e)$ pointing outwards, as shown in Fig. 3 (b). The size of the descriptor is determined with respect to the boundary length and limited to 40 pixels.



(a) (b)

Figure 3. (a) One T-junction includes three segments (A, B, C) and three boundaries $(e_1, e_2, e_3$, in dash blue line). One segment is in front of the other two (A is in front of B and C), and one edge is behind the other two correspondingly (e_2 is behind e_1 and e_3). (b) A vector $\vec{v}(e_3)$ pointing outwards is fit to the boundary e_3. Then an oriented-SIFT descriptor is computed in align with $\vec{v}(e_3)$.

In order to learn the intrinsic appearance of a junction, we use two types of images for this feature: the original image $f_o(e)$ and the binary edge image $f_b(e)$. The binary edge image is a blank image with only the occlusion boundaries labeled in white. While $f_o(e)$ can capture a junction's appearance in the natural image, $f_b(e)$ excludes all the luminance and texture information from the environment, focusing on the boundary distribution within a junction.

We concatenate the above three sets of features as the final junction feature set: $f_j(e) = [f_a(e), f_o(e), f_b(e)]$. Within one junction, the boundaries in front are labeled as $y = 1$ and the boundary behind is labeled as $y = -1$. Then a SVM classifier h_j is trained. During testing, as there is one and only one behind boundary in a valid junction, we enforce this constraint by choosing the behind boundary as the one with the minimum prediction value.

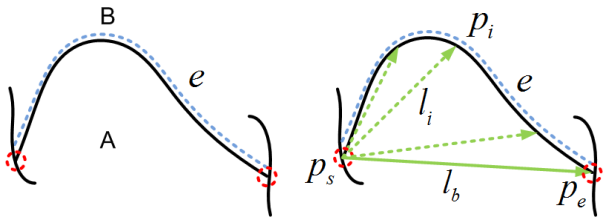
3.2. Boundary feature

In addition to junctions, boundaries are also important for depth ordering. Hoiem et al. [7] proposes local features $f_d(e)$ to encode many edge attributes, and we include them as a subset of our boundary features [1].

Additionally, we consider the boundary convexity an informative clue. Take Fig. 4 (a) as one example, the convexity of boundary e implies that segment A occludes segment B, and thus determines the depth ordering.

Therefore we design features to explicitly capture the boundary convexity. First, we connect the starting point p_s and the ending point p_e of a boundary, and form the base vector l_b. Then the distribution of each point p_i on the boundary with respect to l_b provides the convexity information. Thus we connect every point p_i along the boundary to p_s, and form a new vector l_i. We record the angle between l_i and l_b: $\theta_i = \arccos(l_i \cdot l_{base}) \in [-\pi, \pi]$, as shown in Fig. 4

[1] To follow the convention in this paper, we exclude the high-level geometric context features, which are not applicable for the settings.



(a) (b)

Figure 4. The boundary convexity feature: (a) one occlusion boundary (e.g., e) lies in between two segments (e.g., A, B). Boundary e bends towards segment B, indicating that more likely A is in front of B. (b) A base vector l_b can be set by connecting the two ends p_s and p_e. For each point p_i on the boundary, we link p_s and p_i to create a new vector l_i, and record the angle between l_b and l_i. We histogram these angles as new features for e.

(b). After getting $\{\theta_i\}$ for all $\{p_i\}$, we quantize $[-\pi, \pi]$ into 36 bins and histogram $\{\theta_i\}$, and append this histogram as the new feature $f_c(e)$ in addition to $f_d(e)$: $f_b = [f_d, f_c]$. Since now the boundary is directed from p_s to p_e, for training we label the boundary $y = 1$ if its left segment is in front of its right segment, and $y = -1$ otherwise. Following the same rule, we retrieve the depth ordering of segments during testing.

3.3. Combined features

Junction and boundary features alone have their own strength and weakness, and we combine them together to complement with each other. Since the features in each junction $f_j(e)$ are already computed on the basis of the boundary within it, we can append $f_b(e)$ to $f_j(e)$ to form the combined feature $f_c(e) = [f_j(e), f_b(e)]$.

Accordingly, the learning process on the junction now becomes a ranking problem on the three boundaries/segments. We use a structured SVM [8] $h_c(f_c(e))$ to solve it. For example, in Fig. 3, we can first associate each boundary to the segment on its left. Suppose the ground truth depth order is $A \to C \to B$. Then for boundaries it is $e_3 \to e_2 \to e_1$. During training, the constrains become $h_c(A) > h_c(C)$ and $h_c(C) > h_c(B)$, i.e. $h_c(e_3) > h_c(e_2)$ and $h_c(e_2) > h_c(e_1)$. (We omit f_c for brevity, and in the following we use segment instead of boundary to indicate the depth order, since they are identical.) During testing, let $x_i = \{f_c(A), f_c(B), f_c(C)\}$ be the combined feature on junction i , and y_i^{ABC} be the segment order $A \to B \to C$, then we define the likelihood li of assigning the depth order y_i^{ABC} from the SVM margin:

$$li(y_i^{ABC}|x_i) = \sum_{(M,N)} h_c(f_c(M)) - h_c(f_c(N)), \qquad (1)$$

where $(M, N) \in \{(A, B), (B, C), (A, C)\}$.

4. Towards global depth reasoning

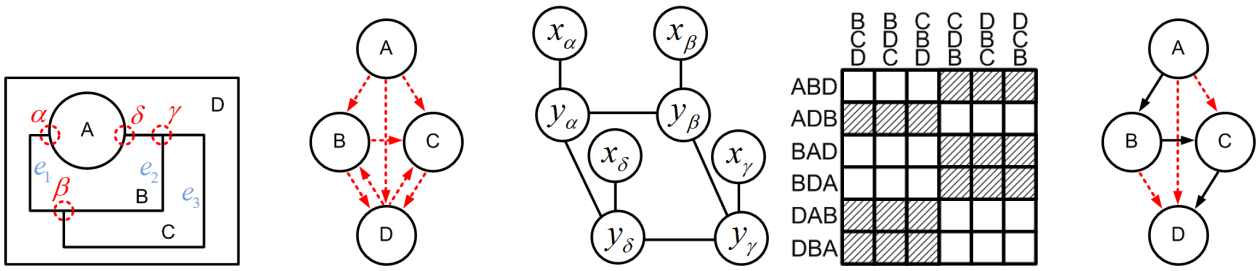
D-order graph: After the local inference for depth ordering, a depth order graph is built (**d-order** graph), shown in Fig. 5 (b), and we assign the depth order for each segment according to this graph. One node in the d-order graph represents one segment in the image. The directed edge indicates one segment is in front of another. With the combined feature f_c, each junction will order its three segments in depth. For example, junction α in Fig. 5 (a) may infer the depth ordering $A \to B \to D$, and produce three directed edges in the d-order graph: $A \to B$, $A \to D$ and $B \to D$.

However, relying on local decision can lead to invalid configuration of d-order graph. Take Fig. 5 (a) as one example: if junction γ incorrectly predicts the order as $D \to B \to C$, while the others have the correct classification, it will introduce a contradiction. This results in a loop of nodes B, C, D in the depth order graph, and makes us impossible to determine the depth order. To solve this problem, we propose a new approach based on Markov Random Field to encourage a more global consistency.

Global: We treat each junction in the image as one node in our MRF graph, shown in Fig. 5 (c). The label space for each node y_i is the possible order permutation of the segments, e.g. for junction α, its y_α will have 6 possible labels of the segment orders: ABD,ADB,...,DBA. The node potential $\phi(y_i|x_i)$ is calculated by taking the negative of Eq.1. The edge in our MRF is defined by the boundary. We link two junctions if they are connected by a boundary in the image. Also if two junctions are connected by a boundary, they must share at least two segments that this boundary separates. Therefore the edge potential $\psi(y_i, y_j)$ is defined as the consistency between the segments' orders.

For instance, in Fig. 5 (a), junction α and β are linked by boundary e_1 (in light blue), and thus α and β share segment B and D that e_1 separates. Accordingly, the segment order on both junctions must be consistent, e.g. the order $A \to B \to D$ on junction α is consistent with the order $B \to C \to D$ on junction β, but the same order for α is inconsistent with the order $C \to D \to B$ on β, because the relative orders of B and D contradict. We build the edge potential $\psi(y_i, y_j)$ following this intuition: we assign zero penalties for the consistent orders, and high penalties for the inconsistent ones. Fig. 5 (d) gives an example of the potential matrix on the edge between node α and β in the MRF, with solid squares representing high penalties.

We use Tree Reweighted Decomposition (TRW) to minimize the total energy function $E = \sum_i \phi(y_i|x_i) + \sum_{i,j} \psi(y_i, y_j)$ for this MRF. Because of the penalties for the inconsistent orders, this optimization process encourages the consistent orders in a more globally optimized manner. Beliefs from other segments are passed through messages to



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

Figure 5. (a) Global depth reasoning example. (b) Each junction produces three directed edges in the depth order graph, e.g. junction α produces the directed edges $A \rightarrow B$,$B \rightarrow D$, and $A \rightarrow D$. (c) We use MRF to encourage the global consistency. Each node corresponds to one junction, and is connected with its neighbors. (d) The edge potential in our MRF gives high penalties (solid) if the segments' orders contradict between two nodes. (e) The depth ordering is assigned by the longest path in the final depth order graph (shown in solid arrow), from which we retrieve the depth ordering, such as $A \rightarrow B \rightarrow C \rightarrow D$.

help local decisions. In practice the inference process usually produces a consistent depth ordering, which enables us to trim the loop in the depth order graph more safely. After that, we find the longest path in the depth order graph (now it is acyclic), and use this path as the skeleton for depth ordering, as shown in Fig. 5 (e). All the other nodes that are not in this skeleton path are assigned with depth values according to this path.

5. Occlusion boundary with closed loops

Segmentation is a necessary preparation for the depth ordering task, and we rely on the occlusion boundary detection to generate it: firstly a dense segmentation using watershed is performed to extract all the possible edges. Then each edge is classified as an occlusion boundary or not. After that the object segmentation is achieved by merging the regions where a non-occlusion boundary lies in between. Our detailed approach is presented as follows.

BoW features: In addition to [7], we propose new features based on bag-of-words [9] for occlusion boundary detection, for they effectively capture the texture information. Each edge from the initial segmentation lies in between two segments. We compute the dense SIFT words within these segments, and histogram them as the new features. Besides, the edge appearance itself provides rich information. If the edge is shaky with noisy curvatures, it is unlikely to be an occlusion boundary. Therefore we also histogram the dense SIFT words along each edge. Together they form the new features for the occlusion boundary detection.

Enforcing the closed loop: Furthermore, occlusion boundaries are not independent. They usually enclose one object and form a closed loop, even when the object is occluded by others. For example, in Fig. 5 (a) segment C is enclosed by edge e_2 and e_3, which together form a closed loop, even though e_2 belongs to segment B. Also the edges inside a loop are less likely to be actual occlusion boundaries.

We explicitly model this property as follows: first we classify each edge and get its belief for the occlusion boundary, shown in Fig. 6 (a). Since each edge connects two junctions at its two ends, we gradually group these junctions to retrieve the loop: initially, each junction in the image forms an individual group. Then we sort all the edges by their predicted beliefs for the occlusion boundary in descending order. After that, we examine each edge from the top belief and its two junctions: if they belong to different groups, we merge them. Otherwise, we find a closed loop with the current maximum predicted belief. If the loop has the size L larger than a minimum requirement L_{min}, we set the beliefs for all the edges l that form the loop as $b_{new} = \sum_l b_l/L$, and lower beliefs by T of the edges connected inside to this loop. The algorithm stops until we examine all the edges with beliefs larger than B_{min}. We also enforce the long edges in a similar way: we group the neighboring edges if they share similar directions, and enhance their beliefs for the occlusion boundary if the group size is large enough.

(a) (b) (c)

Figure 6. (a) The local occlusion boundary detection result (best view in color). Heat map indicates the beliefs for the occlusion boundary, and the redder the higher. (b) We gradually examine the edges with high beliefs and retrieve the loop. (c) We lower the beliefs of the edges that connected inside to this loop.

6. Experiments

We experiment on three different datasets: a synthetic dataset (**syn**), the occlusion boundary dataset provided in [7] (**occ**), and our depth order dataset (**d-order**). Quantitatively we evaluate the depth ordering results by the ordering accuracy: for any two neighboring segments in the image, we examine whether their depth orders are correctly labeled comparing to the ground truth. We compare our final depth

ordering algorithm (**Global**) with the following approaches:

BF: uses the boundary features proposed in [7].

JA: We re-implement the algorithm proposed in [13] that orders the depth mainly by angles within a junction.

pBF: uses the proposed boundary features.

pJF: uses the proposed junction features.

Com: uses the combined the features. The above methods share the same depth reasoning in [13] that deletes the loop in the depth order graph by the lowest local predicted belief.

Global: This is our full algorithm. We use the combined features in **Com** and the proposed MRF graph model to ensure the global depth consistency.

We color each segment by its depth order in the image to visually display the results. Segments in front are darker (more black), and occlude the segments that are brighter (more white). Note that since we don't estimate the absolute depth, but the relative depth order, the absolute color value does not hold a specific meaning. The relative color between segments is more important. Segments are marked by a red "x" if incorrectly labeled in the depth ordering[2].

Generating the object segments is a key step that precedes depth ordering. Since we rely on the occlusion boundary detection to generate the segmentation, we also quantitatively evaluate the average precision for different occlusion boundary detection algorithms. We compare our proposed algorithm **loop** with the following approaches:

bfeat: uses the low-level boundary features from [7] [3].

pfeat: uses the proposed BoW features in addition to **bfeat**.

graph: uses **pfeat** and a graph model (MRF) to enforce the continuity of occlusion boundaries, similar to [7].

loop: This is our full algorithm that uses **pfeat** and explicitly enforces closed-loops and long edges.

Synthetic dataset: We synthetically create a dataset to evaluate the depth ordering algorithms. For this dataset, we randomly place 6 to 10 abstract segments in a image, including rectangles, circles, ellipses etc, with different colors and sizes. Shapes placed later will overlay the previous ones, and in this way we get the ground truth ordering. Examples are shown in 7. We generate 2000 synthetic images, and use half of them for training the depth ordering algorithms, and the other half for testing.

This dataset has perfect segmentation, which enables us to directly compare the performance of different depth ordering algorithms. The depth ordering accuracies are presented in Table 1. The new features on boundaries (**pBF**) and junctions (**pJF**) improve around 3% in accuracy over the baseline feature sets (**BF** and **JA**), showing the effectiveness of our proposed features. Also combining them together (**Com**) achieves better performance over the individual feature set (10% over **pBF** and 4% over **pJF**). Our final algorithm (**Global**) has a clear advantage comparing to all the baselines. Overall **Global** achieves around 10% improvement over the previous works **BF** and **JA**.

[2] In some cases, "incorrect depth" is a relative term between two segments, and we arbitrarily mark one of them.

[3] To follow the convention in this paper and make a fair comparison, in this step we do not compare with the result from the high-level geometric context labels, which are also often inapplicable in the settings.

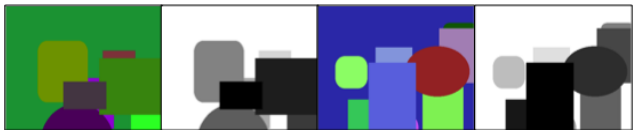
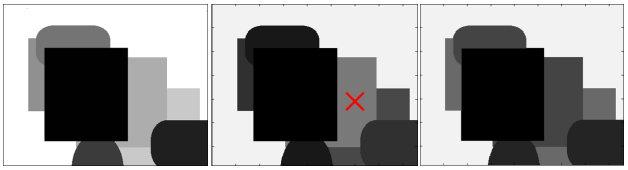


Figure 7. Examples of our synthetic dataset: color images are on the left and the ground truth depth orders are on the right, colored as the front segments are darker.

(a) (b) (c)

Figure 8. (a) Result from **Com**. Combined features can correctly label the depth order (darker segments are in front). (b) Results from **pJF**. (c) Results from **pBF**. Segments are marked by x if incorrectly labeled in the depth ordering.



(a) (b) (c)

Figure 9. Our global reasoning algorithm can provide better depth ordering especially in complicate scenarios. (a) The ground truth depth ordering. (b) Result from **Com**. (c) Result from **Global**. Incorrectly labeled segments are marked by x.

Fig. 8 illustrates the advantage of the combined features. With only the junction features, we cannot infer the depth order between the two rectangles, since their potential junction that can give the right depth order has been blocked, and the result is shown in Fig. 8 (b). On the other hand, using only boundary features makes it impossible to determine the depth order between the ellipse on the top and the rectangle below it, since the boundary in between is a straight line, and the result is shown in Fig. 8 (c). However, when combining these two features, we can correctly label the depth ordering of this image, as shown in Fig. 8 (a).

Our proposed **Global** algorithm outperforms the baselines, especially in the complicate cases when a segment interacts with multiple neighbors. Fig.9 shows one example that when **Global** (shown in (c)) gives in a better depth ordering than **Com** (shown in (b)). The incorrectly labeled segment has four junctions with the segment behind it, and

Figure 11. Example images from our depth order dataset

they may produce inconsistent prediction. However after using the proposed model to enforce the consistency, we can produce a corrected depth order graph.

Occ dataset: We also experiment on the occlusion boundary dataset from [7]. This dataset includes 100 outdoor images with human-labeled segments and their quantized depth. For these natural images, object segmentation is the first step before depth ordering. Therefore two types of experiments are conducted: 1) we order the depth of manually-labeled ground-truth segments (**-gt**). 2) We automatically segment the image by using the occlusion boundary detection result, and then perform depth ordering (**-auto**)[4] [5]. We use 50 for training the occlusion boundary classifier and the depth ordering algorithms, and the other 50 for testing.

Table 1 shows the accuracies in depth ordering on occ dataset, and example results are presented in Fig. 10. Since the variance in this dataset is large comparing to the limited number of training samples (only 50), the margins of the proposed algorithms over the baselines are smaller. However, still **pBF** and **pJF** outperform the baseline features **BF** and **JA** by 1.5% and 3%. **Com** further improves the result by 1%, and **Global** produces the best result.

For generating the segmentation, we show the average precision of the occlusion boundary detection in Table 2. The proposed BoW features give a 5% boost in detecting the occlusion boundary. Enforcing the closed loop (**loop**) marginally outperforms the baseline that uses the graph model (**graph**) and locally enforces the continuity. We believe the small increase is because this dataset is quite challenging. The output occlusion boundary result from the low-level feature **pfeat** is not reliable enough, and thus enforcing the loop may not be significantly better.

D-order dataset: Furthermore, to evaluate the depth ordering algorithms on natural images, we collect a new depth order (d-order) dataset. Various daily objects are placed to occlude each other in different configurations and scenarios. The dataset includes 1087 images. Each object is manually segmented, and its depth is acquired by using the Kinect sensor. Exemplar images are shown in Fig. 11.

We also use half of them for training and the other half for testing, and conduct two experiments: depth ordering on the ground-truth segmentation (**-gt**), and automatically generated segmentation from the occlusion boundary detection (**-auto**). The ground truth depth order of each segment (from either human-labeled or auto-generated) is achieved by averaging the depth values within this segment.

[4]The ground truth depth of each segment from the auto-segmentation is achieved by averaging the depth value over all the pixels in the segment.

[5]for this experiment only, geo-context information provided in [7] is necessary in order to generate usable segmentation.

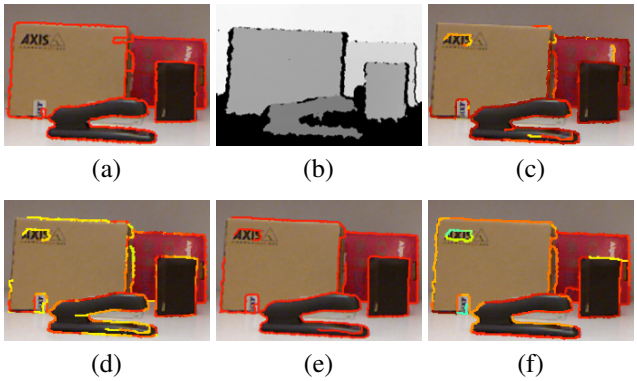


Figure 12. Occlusion boundary detection result (best view in color): (a) the ground-truth occlusion boundary. (b) Depth image from Kinect. (c) Occlusion boundary detection result from **bfeat**. Red in color indicates higher beliefs for the occlusion boundary. (d) to (f) are results from (d) **pfeat**, (e) **graph**. and (f) **loop**.

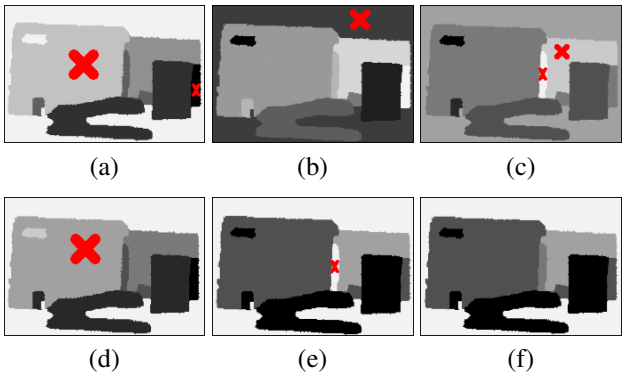
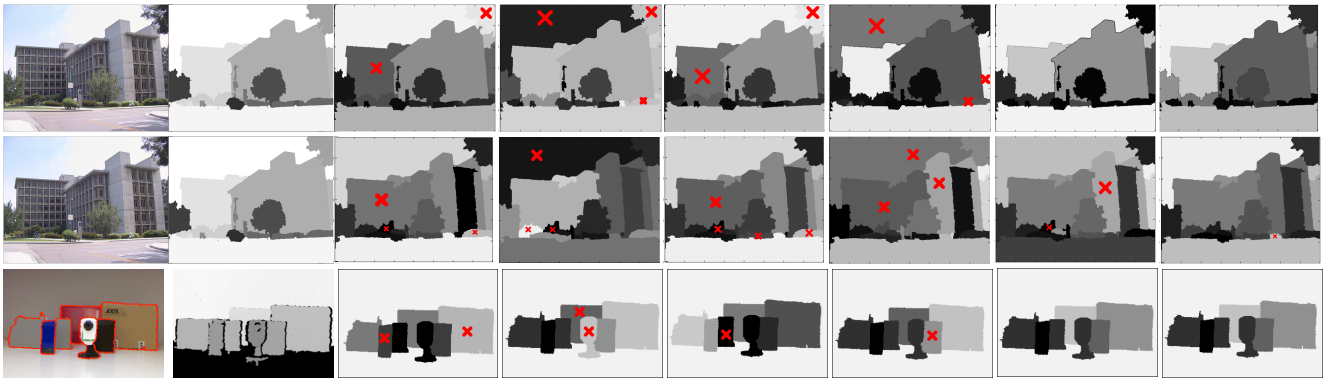


Figure 13. Depth orderings from auto-segmentation. (a) to (f) are results from (a) **BF**, (b) **JA**, (c) **pBF**, (d) **pJF**, (e) **Com**, (f) **Global**.

Table 1 shows the depth ordering accuracy. The new features improve the performances from 1% to 3% over the baselines, and the combined features (**Com**) additionally boosts at least 4% in accuracy. **Global** gives the best performances in all the scenarios, achieving 10% improvement over the previous works in some cases. Fig. 10 and Fig. 13 show the ordering results.

Table 2 presents the average precision of the occlusion boundary detection, and Fig. 12 shows the example results. Our proposed new features outperforms the previous work by 7%, and our final algorithm (**loop**) produces additional 3% higher average precision comparing to the conventional graphical model (**graph**). More importantly, since our algorithm explicitly encourages the loop, it generates more reliable object segmentation for depth ordering.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 10. Results from occ datasets with ground-truth segmentation (**top row**), auto-segmentation (**middle row**), and from our depth order dataset with ground-truth segmentation (**bottom row**). (a) Input image. (b) Ground-truth segmentation and depth. (c) to (h) are results from different methods: (c) **BF**. (d) **JA**. (e) **pBF**. (f) **pJF**. (g) **Com**. (h) **Global**. Incorrectly labeled segments are marked by a red x.

Table 1. Average depth ordering accuracy (in %) of different methods on synthetic dataset (**syn**), occ dataset (**occ**), and our new depth order dataset (**d**). "-gt" : depth ordering is performed on the ground-truth segmentation. "-auto": the segmentation is auto-generated by the occlusion boundary detection.

	BF	JA	**pBF**	**pJF**	**Com**	**Global**
syn	81.0	86.6	83.0	89.9	93.7	**95.4**
occ-gt	70.9	63.3	72.4	66.9	73.2	**73.3**
occ-auto	66.4	58.2	69.5	64.5	69.4	**71.9**
d-gt	82.3	72.5	83.4	75.5	89.2	**91.7**
d-auto	75.0	62.2	75.3	68.3	79.3	**80.3**

Table 2. Average precision (in%) for the occlusion boundary detection on occ dataset (**occ-ap**) and our depth order dataset (**d-ap**).

	bfeat	**pfeat**	graph	**loop**
occ-ap	51.7	57.0	58.3	**58.6**
d-ap	65.5	73.0	75.7	**78.3**

7. Conclusion

We present a learning based framework for depth ordering. We exploit new features on boundaries and junctions, and integrate them to a better feature set for depth ordering. Furthermore, we propose a graph-based algorithm to enforce the global consistency in the depth ordering. We modify occlusion boundary detection algorithm to favor closed loops so that it is better suited for the ordering task at hand. We also collected a new dataset for the depth ordering task. Experiments in various scenarios show our proposed algorithms achieve better performances than the baselines.

For future work, we can further study how the depth ordering helps with segmentation. Besides, we can employ our algorithm in tasks such as object recognition and scene understanding.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. In *CVPR*, 2009. 1

[2] H. G. Barrow and J. M. Tenenbaum. Retrospective on "interpreting line drawings as three-dimensional surfaces". *AI*, 59, 1993. 1

[3] F. Cole, K. Sanik, D. DeCarlo, A. Finkelstein, T. Funkhouser, S. Rusinkiewicz, and M. Singh. How well do line drawings depict shape? *ACM Transactions on Graphics*, 28(3), Aug. 2009. 1

[4] M. Dimiccoli and P. Salembier. Exploiting T-junctions for depth segregation in single images. In *ICASSP*, 2009. 2

[5] D. Hoiem, A. A. Efros, and M. Hebert. Closing the loop in scene interpretation. In *CVPR*, 2008. 1

[6] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. *IJCV*, 80(1), 2008. 1

[7] D. Hoiem, A. A. Efros, and M. Hebert. Recovering occlusion boundaries from an image. *IJCV*, 91(3), 2011. 2, 3, 5, 6, 7

[8] T. Joachims. Making large–scale SVM learning practical. In *Advances in Kernel Methods — Support Vector Learning*. MIT Press, 1999. 4

[9] F. F. Li and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR*, 2005. 1, 5

[10] B. Liu, S. Gould, and D. Koller. Single image depth estimation from predicted semantic labels. In *CVPR*, 2010. 1, 2

[11] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2), 2004. 3

[12] B. Packer, S. Gould, and D. Koller. A unified contour-pixel model for figure-ground segmentation. In *ECCV*, 2010. 1

[13] G. Palou and P. Salembier. Occlusion-based depth ordering on monocular images with binary partition tree. In *ICASSP*, 2011. 2, 6

[14] X. Ren and C. Gu. Figure-ground segmentation improves handled object recognition in egocentric video. In *CVPR*, 2010. 2

[15] X. F. Ren, C. C. Fowlkes, and J. Malik. Figure/ground assignment in natural images. In *ECCV*, 2006. 2

[16] A. Saxena, M. Sun, and A. Y. Ng. Make3D: Learning 3D scene structure from a single still image. *PAMI*, 31(5):824–840, 2009. 1, 3

[17] P. Sundberg, T. Brox, M. Maire, P. Arbelaez, and J. Malik. Occlusion boundary detection and figure/ground assignment from optical flow. In *CVPR*, 2011. 2

[18] D. Waltz. Generating semantic descriptions from drawings of scenes with shadows. Technical Report AI271, MIT, 1972. 1