3D-Based Reasoning with Blocks, Support, and Stability

Zhaoyin Jia[†], Andrew Gallagher[†], Ashutosh Saxena[*], Tsuhan Chen[†]
[†] School of Electrical and Computer Engineering, Cornell University.
[*] Department of Computer Science, Cornell University.
zj32@cornell.edu, andrew.c.gallagher@gmail.com, asaxena@cs.cornell.edu, tsuhan@ece.cornell.edu

Abstract

3D volumetric reasoning is important for truly understanding a scene. Humans are able to both segment each object in an image, and perceive a rich 3D interpretation of the scene, e.g., the space an object occupies, which objects support other objects, and which objects would, if moved, cause other objects to fall. We propose a new approach for parsing RGB-D images using 3D block units for volumetric reasoning. The algorithm fits image segments with 3D blocks, and iteratively evaluates the scene based on block interaction properties. We produce a 3D representation of the scene based on jointly optimizing over segmentations, block fitting, supporting relations, and object stability. Our algorithm incorporates the intuition that a good 3D representation of the scene is the one that fits the data well, and is a stable, self-supporting (i.e., one that does not topple) arrangement of objects. We experiment on several datasets including controlled and real indoor scenarios. Results show that our stability-reasoning framework improves RGB-D segmentation and scene volumetric representation.

1. Introduction

3D reasoning is a key ingredient for scene understanding. Confronted with an image, a human perceives a 3D interpretation of the scene rather than viewing objects as groups of 'flat' color patches. Objects occupy volumes in space, are supported and stable (in static scenes), and occlude farther objects. All these properties require reasoning beyond traditional object recognition.

In this paper, we propose a framework for reasoning with 3D volumes that incorporates the physical constraints of our natural world. Our algorithm inputs RGB-D data, performs 3D box fitting of proposed object segments, and extracts box representation features for scene reasoning, such as box intersection and stability inference.

Past works for producing 3D interpretations represent the world as a "pop-up" model, as piece-wise planar segments, or as blocks constrained to rest on the ground. However, inferring a 3D interpretation is only part of the picture, a good scene interpretation should also follow physical rules: objects should be placed stably in the scene. If we attempt to segment the scene purely based on appearance or shape, we may end up with the segmentations that do not make physical sense, as shown in Fig. 1c. Reasoning about stability brings physics into our model, and encourages more plausible segmentations and block arrangements (see Fig. 1d).

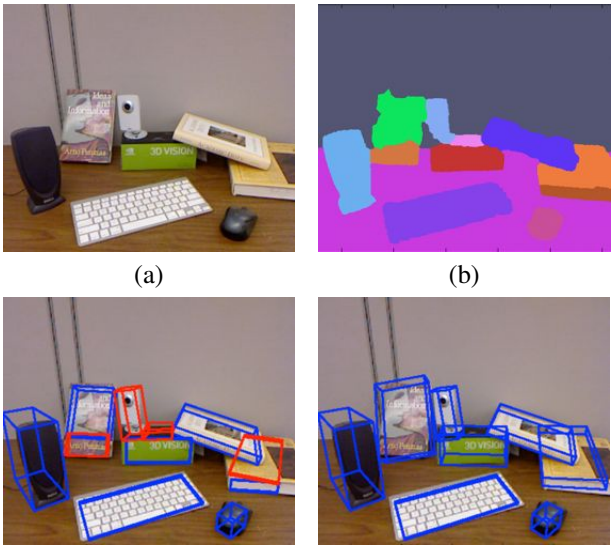


Figure 1. (a) The input RGB-D image. (b) Initial segmentation from RGB-D data. (c) A 3D bounding box is fit to the 3D point clouds of each segment, and several features are extracted for reasoning about stability. Unstable boxes are labeled in red. (d) The segmentation is updated based on the stability analysis and it produces a better segmentation and a stable box representation.

The challenge is that objects can be arranged in complicated configurations. While some recent work considers notions of support (e.g., [9, 14, 23]), they are limited to single support or isolated objects on a flat surface. Thus, these methods do not apply to more complicated stacking arrangements of objects that can occur, for example, on

desks or other cluttered situations.

In detail, we first fit a 3D box to the point-cloud of each segment, and then extract several features to reason about the scene: 1) we define the box fitting error based on the 3D points and box surfaces; 2) we ensure that 3D points lie on the visible surfaces of the boxes given the camera position; 3) we find space violations when neighboring boxes intersect one another; 4) we propose supporting relations and the stability of the scene given the boxes. This evaluation of the box representation allows us to refine the segmentation based on these box properties through a learning process.

We experiment on several datasets, from a synthetic block dataset to the NYU dataset of room scenes, and a new Supporting Object Dataset (SOD) with various configurations and supporting relations. Experimental results show that our algorithm improves RGB-D segmentation. Furthermore, the algorithm provides a 3D volumetric model of the scene, and high-level information related to stability and support.

To summarize, our major contributions are:

1. A volumetric representation of the RGB-D segments using boxes.
2. Novel features based on box representation and stability reasoning.
3. A learning framework for inferring the object segmentation in the scene.
4. A new supporting objects dataset including human segmentation and support information.

2. Related work

Geometric inference from a single color image has been investigated in [11] and [21, 22] for estimating the depth of each segment using only color features. The results appear either in the format of "pop-up images" [5]: segments stand like billboards in different depth layers, and have empty space behind, or as piecewise planar segments [22]. The limitation is obvious: these models do not align with our understanding of the scene, where each object actually occupies some space in 3D, as we do in this work (Fig. 1d).

To overcome this limitation, Gupta et al. [9] proposed a block world representation to fit 2D color segments. Segments in outdoor scenes are represented by one of eight predefined box types that represent a box viewed from various positions. Although the buildings in these outdoor scenes often fit nicely into one of the block categories, this assumption is not true for general images of stacked objects, where the orientations of objects are not limited to eight. Zheng et al. [25] also used blocks representation for objects, but required interactive human labelings for non-box objects. Xiao et al. [24] propose detecting 3D cuboids solely in RGB images, and Bleyer et al. [2] show box fitting for improved stereo. In this work, we use RGB-D data and fit boxes with depth information for volumetric and stability reasoning.

In addition, researchers have studied indoor environment reasoning on color images, where the 3D geometric inference can be approximated as a Manhattan World [6] [10] [18]. Indoor images have strong clues of lines and planes as well as a fixed composition of ceiling, wall and ground. However, these approaches will likely to fail if only partial or no wall/floor are captured, and at a close-up views of small objects lying on the table.

Previous work has shown that integrating depth with color information improves many vision problems, such as segmentation [23], object recognition ([12], [13] and [17]), scene labeling [15], and activity detection [16]. These algorithms usually treat depth as another information channel without explicitly reasoning about the space that an object occupies. For example, if one object is partially observed, it remains hollow inside. In this way, segmentation and supporting inference are transformed into a classification problem in a 2.5-D space. In contrast, we explicitly reason about full 3D models by fitting boxes to objects. This leads to a more natural interpretation of the scene, provided by better segmentation and support inference.

Grabner et.al. [8] analyzes the interaction between human and objects such as chairs in 3D space. The algorithm finds the object support, and shows that a 3D model can predict well where a chair supports the person. This also helps chair detection. However, in this paper, we perform a more general analysis of the 3D objects in the scene through box fitting and stability reasoning.

Jiang et al. [14] reason about stability for object arrangement, but their task is different from ours: given a few objects, their goal is to *place* them in the environment stably. In other recent works, Silberman et al. [23] identify which image segments support which other segments. However, reasoning about support and stability are two different things. Past work on support pre-supposes that segmentations are already stable, and implicitly assumes that all regions need only one region to support them, without checking any physics-based model of stability. We use stability reasoning to verify whether a given volumetric representation of a scene could actually support itself without toppling, and adjust the segmentation accordingly.

We use a simple model for evaluating the stability of our block arrangements, although more complicated physics-based simulators [1] could be employed. One approach could be to consider all possible reasonable segmentations, and plug each into a simulator. However, this would result in an exponential number of evaluations, and would still be susceptible to noise and other unknown physical parameters (e.g., coefficients of friction). Our approach for stability evaluation is based on a simple Newtonian model: the center of gravity of each adjacent object subset must project within its region of support. This simple model is justified by the ideas of intuitive physics [20] that humans even have

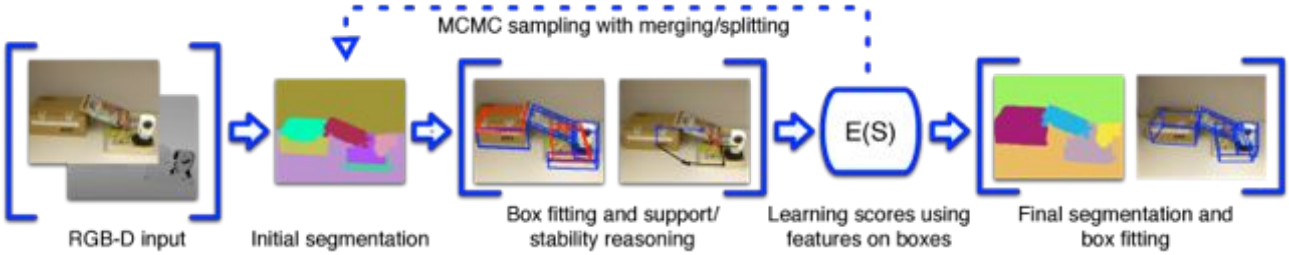
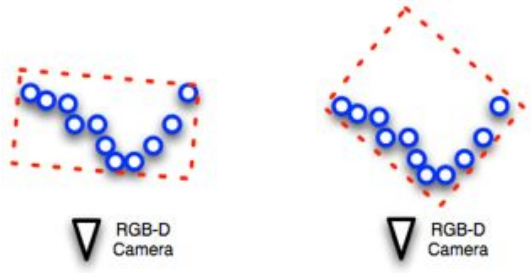


Figure 2. An overview of our algorithm.



(a) (b)

Figure 3. (a) A bounding box fit based on minimum volume may not be a good representation for RGB-D images, where only partially observed 3D data is available. (b) A better fit box will not only enclose a small volume, but also have many points near the box surface. Data points are projected in 2D for illustration.

a sense of stability at a glance. Our algorithm is not a perfect reflection of the physical world, but it is accurate enough to achieve our goal of improved 3D scene parsing.

3. Approach Overview

Our input is an initial RGB-D over-segmentation. First, we fit a 3D bounding box to each segment in the 3D point-cloud. Next, we compute features between boxes and propose supporting relations, perform stability reasoning, and adjust the box orientation based on the supporting surfaces. Finally, we extract features on the single and pairwise neighboring segments from these 3D boxes, and model the segmentation with an energy function based on learned regressors using these features. Fig. 2 shows an overview.

4. Single box fitting

RGB-D data is observed from only one viewpoint, and fitting 3D bounding boxes with minimum volumes [3] may fail. Fig. 3a gives an illustration. A minimum volume box covers all the data points but might not give the correct orientation of the object, and fails to represent the object well. A well-fit box should have many 3D points near box surfaces, as shown in Fig. 3b.[1]. We use a RANSAC based algorithm (details below) to fit boxes to point clouds.

[1]Recent related work [19] considered cylinder fitting of 3D points to the surface but did not consider visibility.

4.1. Minimum surface distance

The orientation of a 3D bounding box is determined by two perpendicular normal vectors (the third normal is perpendicular to these two vectors). The idea is to find the two principle orientations of the 3D bounding box so that the 3D points are as close as possible to the box surfaces. Given a set of 3D points $\{P_i\}$ and a proposed 3D box, we calculate the distance of each point to the 6 surfaces of the box, and assign each point to this nearest-face distance $\{D_{min}(P_i)\}$. The objective for our box fitting algorithm is to minimize this sum for all the 3D points: $\sum_i D_{min}(P_i)$.

We use RANSAC to find a plane to fit all the 3D points within one segment, and it provides the first surface S_1. Next, we collect the outlier 3D points not near S_1, and project them to surface S_1 as 2D points $\{p_j\}$. Then we retrieve the boundary points $\{p_{j,boundary}\}$ that form the convex hull of $\{p_j\}$, and fit a line to $\{p_{j,boundary}\}$ also using RANSAC. This line gives the second surface orientation perpendicular to S_1.

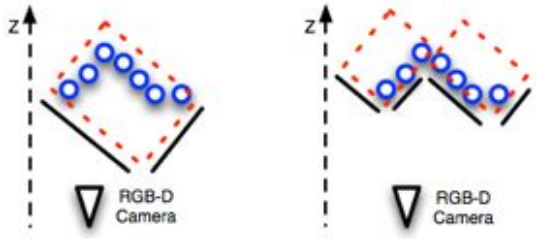
The above steps give the orientations that aligns with many points. The minimum volume is determined by finding the extent of the 3D points given the box orientation. Note that there are usually noisy depth points: If a segment mistakenly includes a few points from other segments before or behind, it can lead to a large increase of the box volume. Therefore, we allow for up to 5% outliers in the 3D points.

With the final 3D bounding box, the sum of the minimum surface distance of the point, $\sum_i D_{min}$, is calculated. The whole process is repeated several times and the best fitting box (smallest distance $\sum_i D_{min}$) is chosen.

4.2. Visibility

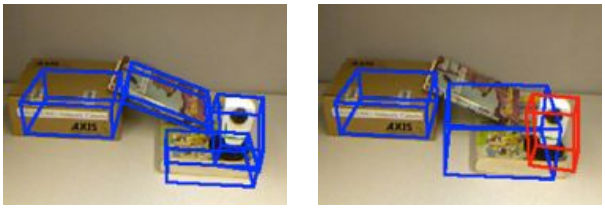
We identify which box surfaces are visible to the camera. If the objects in the scene are mostly convex, then most 3D points should belong to the visible box surfaces instead of hidden faces.

Fig. 4 illustrates the visibility feature for our box fitting. Surface visibility is determined by the position of the camera center and the surface normal. We define the positive normal direction of a surface as the normal pointing away from the box center, and then a surface is visible if the camera center lies at its positive direction. Each box has at most three visible surfaces. We compute the percentage of the



(a) (b)

Figure 4. Given the camera position and a proposed bounding box, we determine the visible surfaces of the box, shown as a solid parallel black line to the box surface. (a) This box may give a compact fit, but most of the points lie on the hidden surfaces. (b) With a better box fit, most of the points lie on the visible surfaces of the two boxes.



(a) (b)

Figure 5. (a) Well fit boxes should not intersect much with neighboring boxes. (b) If two segments are merged incorrectly, e.g., the two books in the image, then the new box fit to the segment is likely to intersect with neighboring boxes, e.g., the box shown in red.

points that belong to visible surfaces, and use this as the feature for later processing.

5. Pairwise box interaction

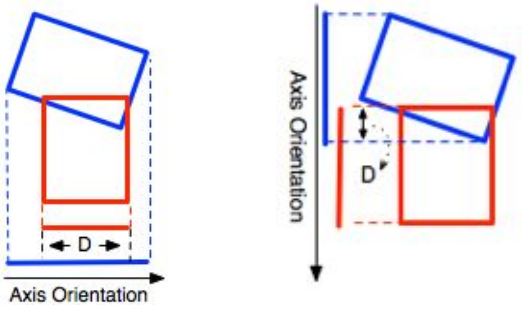
We examine the two pairwise relations between nearby boxes: box intersection, and box support.

5.1. Box intersection

Box intersection gives an important clue for volume reasoning. Ideally, a box fit to an object should contain the object's depth points, and not intrude into neighboring boxes. If a proposed merging of two segments produces a box that intersects with many other boxes, it is likely an incorrect merge. An example is shown in Fig. 5.

We explicitly compute the box intersection, and the minimum separation distance between box pairs and direction. Since 3D bounding boxes are convex, we apply the Separating Axis Theorem (SAT) [7], used in computer graphics for collision detection. We present a 2D illustration of finding the distance of the box intersection in Fig. 6. The distance D shown in Fig. 6b is the minimum moving distance to separate two intersecting boxes.

Extending this algorithm to 3D bounding boxes is straight-forward: since three surface orientations of a box are orthogonal to one another, we examine a plane parallel



(a) (b)

Figure 6. Separating Axis Theorem in 2D: (a) to separate two boxes, we rotate the axis perpendicular to any of the edge, and project all the vertices to this rotated axis. (b) If two bounding boxes are separate, there exists an axis that has a zero overlap distance (D in the image). We examine all the possible axis rotations (in this case four possibilities), and choose the minimum overlap distance. This gives the orientation and the minimum distance required to separate two boxes.

to each surface, and project the vertexes of the two pairwise boxes to this plane. We compute the convex hull of the projection of each box, check whether the two convex hulls intersect and find their minimum separating distance D.

This process gives both separating distance, and the orientation θ_{sep} to separate the two boxes with the minimum distance. θ_{sep} is used when determining the pairwise supporting relations between boxes. For non-intersecting boxes, we choose the orientation and the distance that maximally separate the two boxes as their intersection features.

5.2. Box supporting relation

In order to address various supporting scenarios, we define three supporting relations between the boxes: 1) surface on-top support (and object is supported by a surface from below); 2) partial on-top support (an object is tilted and only partially supported from below); 3) side support. Examples are shown in Fig. 7 (a) to (c).

To classify supporting relations, we detect the ground and compute the ground orientation following [23]. We define the 3D axis as the follows: the xz-plane is parallel to the ground plane, and $y = -1$ is the downward gravity vector. We align the point-cloud with this axis.

Given the box representation of the scene, we classify pairwise supporting relations with the following set of rules: 1) we use the separating orientation θ_{sep} to distinguish between "on-top" support and the "side" support: an "on-top" support has a separating direction nearly parallel to y axis ($< 20°$), while the "side" support has a separating direction close to parallel to the xz-plane (ground plane); 2) for "on-top" supporting relations, there are two possibilities: an even on-top support, shown in Fig. 7a, and a tilted on-top support, shown in Fig. 7b. We distinguish these two types by examining the two closest surfaces of the pairwise

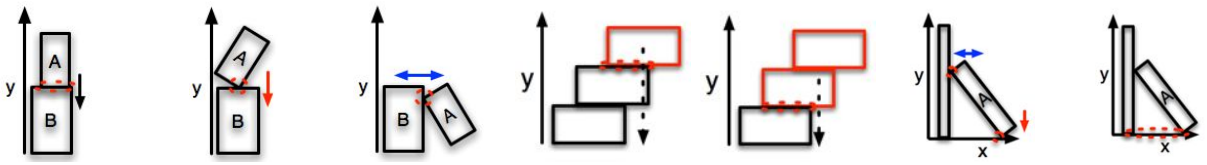


Figure 7. (a) to (c): three different supporting relations: (a) surface on-top support (black arrow); (b) partial on-top support (red arrow); (c) side support (blue arrow). Different supporting relations give different supporting areas plot in red dashed circles. (d) to (e): stability reasoning: (e) considering only the top two boxes, the center of the gravity (in black dashed line) intersects the supporting area (in red dashed circle), and appears (locally) stable. (e) When proceeding further down, the new center of the gravity does not intersect the supporting area, and the configuration is found to be unstable. (f) to (g) supporting area with multi-support: (f) one object can be supported by multiple other objects. (g) The supporting area projected on the ground is the convex hull of all the supporting areas.

boxes. If these two surfaces have a large angle difference ($> 20°$) with each other, and have different orientations to the ground plane, then it is classified as a partial "on-top" support, i.e., the object on top is tilted. Otherwise it is a "surface on-top" support.

Reasoning about stability requires that we compute centers of mass for object volumes, and determine areas of support (i.e., regions or points of the object that are supported, either on side or beneath). Stability requires that the projection of the center of mass of the object along the gravity vector falls within the region of support. We use an object's supporting relation to find the supporting area projected on the ground, and different supporting relations provide different supporting areas. For "surface on-top" support, we project the vertexes of the two 3D bounding box to the ground, compute the convex hull for each projection, and use their intersection area on the ground plane as the supporting area. For "partial on-top" and "side" support, we assume there is only one edge touching between two boxes, and project this touching edge on the ground plane as the supporting area. Examples of the supporting areas are shown as red dashed circles in Fig. 7 (a) to (c).

6. Global stability

Box stability is a global property: boxes can appear to be fully supported locally, but still be in a globally unstable configuration. Fig. 7 (d) and (e) provide an illustration.

We perform a top-down stability reasoning by iteratively examining the current gravity center and supporting areas. This process is shown in Fig. 7. For simplicity we assume each box has the same density.

We begin with the top box by finding the box center of mass, and check whether its gravity projection intersects the supporting area. If so, we mark the current box stable, and proceed to another box beneath for reasoning. Following the constant density assumption, the center of mass $P_c = [x, y, z]$ for a set of boxes is calculated by averaging the volume V_i of each box i:

$$P_c = (\sum_i P_{c,i} \cdot V_i) / \sum_i V_i \qquad (1)$$

We iteratively update the center of mass by adding the boxes below until we reach the ground. If we found that the current supporting area does not support the center of mass, we label the current box unstable, shown in Fig. 7e. For the set of boxes with multiple supports, we compute the convex hull of the multi-supporting areas as the combined supporting area, shown in Fig. 7 (f) to (g).

Support reasoning: Stability reasoning helps delete unnecessary supports. For example, side-to-side *nearly touching* objects do not necessarily support one another. We trim these unnecessary supporting relations by examining the support relations in the order: surface on-top, partial on-top and side support. If the object has a "surface on-top" support and the configuration can be stable, then additional support relations are unnecessary and can be trimmed. If not, then we find a minimum combination of the on-top supports (both surface and partial) and at most two side supports to see if the object can be stable. If so, all other support relations for this object are deleted.

Box fitting: Stability reasoning and supporting relations are used to refine the orientation of a box. If the box is fully supported through a "surface on-top" relation, then we refit the 3D bounding box of the object on top, confining the rotation of the first principle surface S_1 to be the same as the supporting surface. We repeat the supporting relation inference and stability reasoning with the re-fitted boxes. This improves the box representation and support interpretation of the scene.

7. Integrating box-based features

For the segmentation application, we start with some initial segments generated with features from [23]. Then, using the ground-truth segmentation, we label the segments that should be merged as $y = 1$, and the others as $y = 0$. We extract a set of features x based on the box fitting, pairwise box relation, and the global stability, shown in Table 1. For example, for a merge move, we record the minimum surface distances of two neighboring boxes before merging (2 dimensions, noted as **B**), and the minimum surface distance of the box after merging (1 dimension, noted as **A**), as well

Table 1. Features based on volumetric and stability reasoning. **B**: the feature before a move; **A** the feature after a move; **D**: the difference of the feature before and after a move. Possible moves are merging and splitting.

Single/Pairwise features	**dim**
Box orientation with respect to the ground (B, A)	3
mean of the minimum surface distance (B, A, D)	5
Percentage of the visible points (B, A)	3
Percentage increase in the invisible points after a move	1
Number of intersecting boxes (B, A, D)	5
Average intersecting distance of the boxes (B, A, D)	5
Average intersecting distance of the boxes (B, A, D) with respect to volume	5
Pairwise supporting relations	1
Stability features	**dim**
Global stability (B, A, D)	3
Stabilities of the objects (B, A)	3
Distance of the projected gravity center to the supporting area center (B, A, D)	5
Difference for the three supporting relations	3
Average over number segments of the three supporting relations (B, A)	6

as the difference of this criterion before and after merging (1 dimension for each box before merging, 2 dimensions in total, noted as **D**).

As a base model (**Stability**), we train an SVM regression $y = f(x)$ based on these features x and labels y. During testing, we greedily merge the neighboring segments based on the output prediction of the regression f, fit a new bounding box for the segment, perform stability reasoning, and re-extract the features for regression. We repeat the above steps until the classifier does not suggest merging.

8. Splitting and Merging with MCMC

In this Section, we improve our model (**Stability**) from Section 7 by introducing an energy function with unary and pairwise terms based on the volumetric boxes, their support relations, and stability (**MCMC**). We use s_i to represent one individual segment in a segmentation. Given a segmentation $S = \{s_1, ..., s_N\}$ with N segments and M pairs of neighboring segments,[2] we define the energy function:

$$E(S) = \frac{1}{N}\sum_i \phi(s_i) + \frac{1}{M}\sum_{i,j} \psi(s_i, s_j), \qquad (2)$$

where $\phi(s_i)$ is a regression score of a segment s_i describing the quality of the segment when compared with the ground-truth, and it is learned using single box features and its stability. $\psi(s_i, s_j)$ is a regression score of two neighboring boxes learned using pairwise box features and their support relations.

Our goal is to minimize this energy function and find the optimal segmentation S^*: $S^* = \arg\min_S E(S)$. Note that this energy function is non-convex, and the space of possible segmentations $\{S\}$ is very large. However, only a few segmentations are likely. In order to explore the space, we adopt a Markov-Chain-Monte-Carlo (MCMC) [4] approach to this problem, where we design appropriate moves to explore the space. We start with an initial segmentation, and move to a new set of segmentations by either: (a) merging two neighboring segments into one; or (b) splitting one segment into two smaller segments based on the boundary beliefs from [11]. We then re-evaluate the segmentations after making one move, and take the top K ($= 5$ in our setting) segmentations for the next iteration. In practice, this algorithm optimizes our energy function to a reasonable local minima in about 10-15 iterations.

[2] Here $\{S\}$ represents the space of all possible segmentations

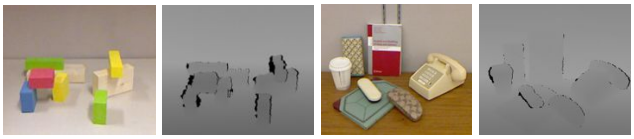


Figure 8. Examples of the RGB-D Block Dataset (left) and Supporting Object Dataset (right).

9. Experiments

We experiment on three datasets: a block dataset, a supporting object dataset, and a dataset of indoor scenes [23].

9.1. Block dataset

We apply our algorithm to a toy block dataset. This dataset has 50 RGB-D images of blocks, shown in Fig. 8, left. For each block, we manually provide the ground-truth segment labels, as well as the orientations of two perpendicular surfaces. Ground-truth surface orientations are labeled by manually clicking at least 8 points on the same surface, and fitting a plane to these labeled 3D points. Supporting relations of each block are also labeled.

First we evaluate our box fitting algorithm. The following algorithms are compared:

Min-vol: the baseline algorithm from [3] of fitting minimum volume bounding box .

Min-surf: the proposed box fitting algorithm of finding the minimum surface distance.

Supp-surf: use our proposed algorithm **Min-surf** to find the initial boxes, and adjust the orientation of the box based on the supporting relations and stability.

We compare the orientation of the bounding box from each algorithm to the ground-truth, and calculate the average angle difference. Table 9.1 shows that our proposed minimum surface distance provides a better box fitting compared to the minimum volume criteria, reducing the errors in angle to 40%. With stability reasoning, the fitting decreases error by another 15%.

We then analyze the performance of our stability reasoning. We compare with the ground truth supporting relations,

Table 2. Average angle error on the bounding box orientation.

Min-vol	**Min-surf**	**Supp-surf**
15.41°	9.75°	7.02°

Table 3. Supporting relation accuracy for different dataset.

	neighbor	**stability**
Block	80.59%	**91.68%**
SOD	52.88%	**72.86%**

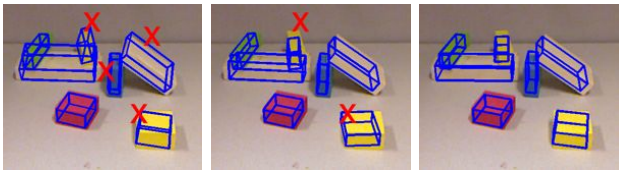


Figure 9. Fitting results on the block dataset. Left, **Min-vol**. Middle, **Min-surf**. Right, **Supp-surf**. Blocks with large fitting error in orientation are labeled as a red "x".

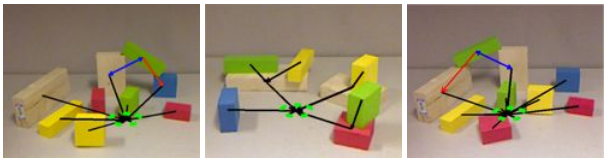


Figure 10. The predicted supporting relations on block dataset. Three different types of the supporting relations are colored in black (surface-top), red (partial-top), and blue (side). The ground plane center is plot as a green dashed circle.

and count an object as correct if all its supporting objects are predicted. We compare our proposed algorithm (**stability**) that reasons about the stability of each block and deletes the false supporting relations with the baseline (**neighbor**) that assumes one block is supported by its neighbors, i.e., the initialization of the supporting relations.

Table 3 reports the supporting relation accuracy. Since the segments in the dataset are perfect blocks, the neighboring rule gives a high accuracy at over 80% for predicting support. However, our proposed stability reasoning improves the supporting relation accuracy by an absolute 10%, achieving over 90% of accuracy. Exemplar images of the predicted supporting relations are shown in Fig. 10.

9.2. Supporting object dataset

We collect a new Supporting Object Dataset (SOD) composing of 307 RGB-D images. Various daily objects are randomly placed in scenes in different configurations of support. For each object, we manually label the segment and the other objects supporting it. See Fig. 8, right, for a few examples.

First, we measure the prediction of the supporting relations with the ground truth segmentation. The results of using the baseline **neighbors** and our stability reasoning **stability** are shown in Table. 3, bottom row. In this dataset with irregular shaped objects and complicated support configurations, using the touching neighbors to infer supporting relations has an accuracy of 52%. Stability reasoning gives an absolute 20% boost, reaching over 72% accuracy. Fig. 12 presents the exemplar results of our box fitting and support prediction from the supporting object dataset.

Table 4. Pixel-wise segmentation score.

	[23]	**S/P**	**Stability**	**MCMC**
SOD	60.2%	64.7%	66.7%	**70.0%**
NYU	60.1%	60.8%	61.0%	**61.7%**

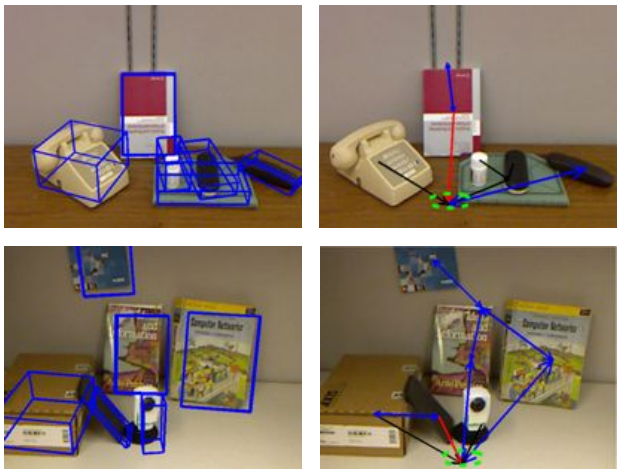


Figure 12. We qualitatively show our box fitting algorithm (left) on daily objects with ground-truth image segmentation and the supporting relation prediction after stability reasoning (right). Boxes for large surfaces (like the back wall and the ground) are not displayed for better visualization. The ground plane is plot as a green dashed circle for showing the support inference results.

We also evaluate the segmentation performance with our proposed features based on box properties. We randomly choose half of the images for training, and the other half for testing. We follow the procedure in [23] and use their color and depth features as the baseline. Then we add our features using the single and pairwise box relations (**S/P**), and our full feature set with stability reasoning (**stability**) with the model proposed in Section 7. Finally we perform our final model based on the energy function with MCMC sampling allowing both merging and splitting (**MCMC**).

The segmentation accuracy is scored by pixel-wise overlapping with the ground-truth segments, proposed in [11] and [23]. Table 4, top row, shows the performance comparison with different feature sets for our proposed dataset (evaluating only on the object segments because the background is shared across the images). Reasoning about each object as a box gives around 4% boost in segmentation accuracy, and adding the stability features further improves the performance by 2%. Our final energy model with MCMC sampling gives the best results with another 3% improvement. Testing results are presented in Fig. 11.

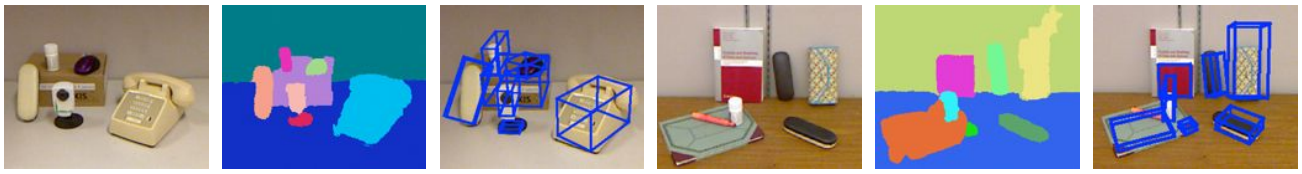


Figure 11. Segmentation and box fitting results of our proposed algorithm on the testing images.

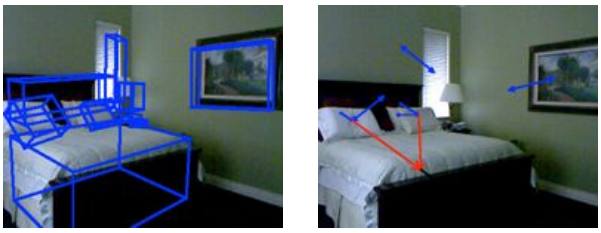


Figure 13. Qualitative result of box fitting (left) and supporting relation inference (right) on indoor scenes. For better visualization, boxes that are too large (wall, ground) or too small are not displayed.

9.3. NYU indoor dataset

We evaluate segmentation performance on the newly released RGB-D NYU indoor dataset [23], and report the performance in Table 4, bottom row. This dataset is proposed for scene understanding instead of object reasoning, and many large surfaces, such as counters and drawers, are not labeled as boxes as the framework proposed in this paper. Although these conditions limit the performance of the proposed algorithm, adding the proposed features improves the performance. We qualitatively present the box fitting and supporting inference result with ground-truth segmentation in Fig. 13.

10. Conclusion

In this paper, we propose analyzing RGB-D images through physical-based stability reasoning. We begin with box fitting on partially observed 3D point clouds, and then introduce pairwise box interaction features. We explore global stability reasoning on proposed box representations of a scene. Segmentations associated with unstable box configurations are not physically possible and are therefore modified. Stability reasoning allows us to improve reasoning about supporting relations (by requiring enough support to provide stability for each object) and improve box orientation (by knowing when objects are fully or partially supported from below). Experiments show that our proposed algorithm works in synthetic scenarios as well as real world scenes, and leads to improvements in box fitting, support detection, and segmentation.

Acknowledgement: We thank Daniel Jeng for useful discussions about stability reasoning. This work is supported in part by NSF DMS-0808864.

References

[1] D. Baraff. Physically based modeling: Rigid body simulation. Technical report, Pixar Animation Studios, 2001. 2

[2] M. Bleyer, C. Rhemann, and C. Rother. Extracting 3D scene-consistent object proposals and depth from stereo images. In *ECCV*, 2012. 2

[3] C. Chang, B. Gorissen, and S. Melchior. Fast oriented bounding box optimization on the rotation group $\mathrm{SO}(3, R)$. *ACM Transactions on Graphics*, 30(5), 2011. 3, 6

[4] J. Chang and J. W. Fisher. Efficient MCMC sampling with implicit shape representations. In *CVPR*, 2011. 6

[5] H. D, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *IJCV*, 75(1):151–172, 2007. 2

[6] A. Flint, D. W. Murray, and I. Reid. Manhattan scene understanding using monocular, stereo, and 3D features. In *ICCV*, 2011. 2

[7] S. Gottschalk. Separating axis theorem. *Technical Report*, 1996. 4

[8] H. Grabner, J. Gall, and L. J. V. Gool. What makes a chair a chair? In *CVPR*, 2011. 2

[9] A. Gupta, A. A. Efros, and M. Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *ECCV*, 2010. 1, 2

[10] V. Hedau, D. Hoiem, and D. A. Forsyth. Recovering free space of indoor scenes from a single image. In *CVPR*, 2012. 2

[11] D. Hoiem, A. N. Stein, A. A. Efros, and M. Hebert. Recovering occlusion boundaries from a single image. In *ICCV*, 2007. 2, 6, 7

[12] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell. A category-level 3-D object dataset: Putting the kinect to work. In *ICCV workshop*, 2011. 2

[13] Y. Jiang, H. Koppula, and A. Saxena. Hallucinated humans as the hidden context for labeling 3d scenes. In *CVPR*, 2013. 2

[14] Y. Jiang, M. Lim, C. Zheng, and A. Saxena. Learning to place new objects in a scene. *IJRR*, 31(9), 2012. 1, 2

[15] H. Koppula, A. Anand, T. Joachims, and A. Saxena. Semantic labeling of 3D point clouds for indoor scenes. In *NIPS*, 2011. 2

[16] H. Koppula, R. Gupta, and A. Saxena. Learning human activities and object affordances from rgb-d videos. *IJRR*, 2013. 2

[17] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view RGB-D object dataset. In *ICRA*, 2011. 2

[18] D. C. Lee, A. Gupta, M. Hebert, and T. Kanade. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *NIPS*, 2010. 2

[19] D. Ly, A. Saxena, and H. Lipson. Co-evolutionary predictors for kinematic pose inference from rgbd images. In *GECCO*, 2012. 3

[20] M. McCloskey. Intuitive physics. *Scientific American*, 248(4):114–122, 1983. 2

[21] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *NIPS*, 2005. 2

[22] A. Saxena, M. Sun, and A. Y. Ng. Make3D: Learning 3D scene structure from a single still image. *PAMI*, 31(5), 2009. 2

[23] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, 2012. 1, 2, 4, 5, 6, 7, 8

[24] J. Xiao, B. C. Russell, and A. Torralba. Localizing 3D cuboids in single-view images. In *NIPS*, 2012. 2

[25] Y. Zheng, X. Chen, M. Cheng, K. Zhou, S. Hu, and N. J. Mitra. Interactive images: cuboid proxies for smart image manipulation. *ACM Trans. Graph*, 31(4):99, 2012. 2