3D Reasoning from Blocks to Stability

Zhaoyin Jia, *Student Member, IEEE,* Andrew C. Gallagher, *Senior Member, IEEE,*
Ashutosh Saxena, *Member, IEEE,* and Tsuhan Chen, *Fellow, IEEE*

Abstract—Objects occupy physical space and obey physical laws. To truly understand a scene, we must reason about the space that objects in it occupy, and how each objects is supported stably by each other. In other words, we seek to understand which objects would, if moved, cause other objects to fall. This 3D volumetric reasoning is important for many scene understanding tasks, ranging from segmentation of objects to perception of a rich 3D, physically well-founded, interpretations of the scene.
In this paper, we propose a new algorithm to parse a single RGB-D image with 3D block units while jointly reasoning about the segments, volumes, supporting relationships and object stability. Our algorithm is based on the intuition that a good 3D representation of the scene is one that fits the depth data well, and is a stable, self-supporting arrangement of objects (i.e., one that does not topple). We design an energy function for representing the quality of the block representation based on these properties. Our algorithm fits 3D blocks to the depth values corresponding to image segments, and iteratively optimizes the energy function. Our proposed algorithm is the first to consider stability of objects in complex arrangements for reasoning about the underlying structure of the scene. Experimental 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. A human perceives and interprets a scene as a collection of 3D objects. Rather than groups of 'flat' color patches, we perceive objects in space with perspective. In static scenes, we understand that objects occupy volumes in space, are supported by other objects or the ground, are typically stable (i.e., not falling down or toppling), and occlude farther objects. These physical properties are usually not considered in traditional object recognition.

In this paper, we propose a framework for 3D segmentation and scene reasoning with volumetric blocks that incorporates the physical constraints of our natural world. Our algorithm takes RGB-D data as input, performs 3D box fitting of proposed object segments, and extracts box representation features (such as box intersection and stability inference) for a physics-based scene reasoning. Our final output is the object segmentation of the scene, and its block representation (shown in Fig. 1 (d)).

Past works for producing 3D interpretations represent the world as point-wise depth-grid [1], as a "pop-up" model [2], as piece-wise planar segments [3], [4], or as blocks constrained to rest on the ground [5]. However, inferring a 3D interpretation is only part of the picture, a good scene interpretation should also follow physical rules: assuming the image captures a static scene, objects should be placed stably. If we attempt to segment the scene purely based on appearance or shape, we may end up with segmentations that do not make physical sense, as shown in Fig. 1 (b). Reasoning about stability brings physics into our model, and encourages more plausible segmentations and block arrangements, such as the example presented in Fig. 1 (d).

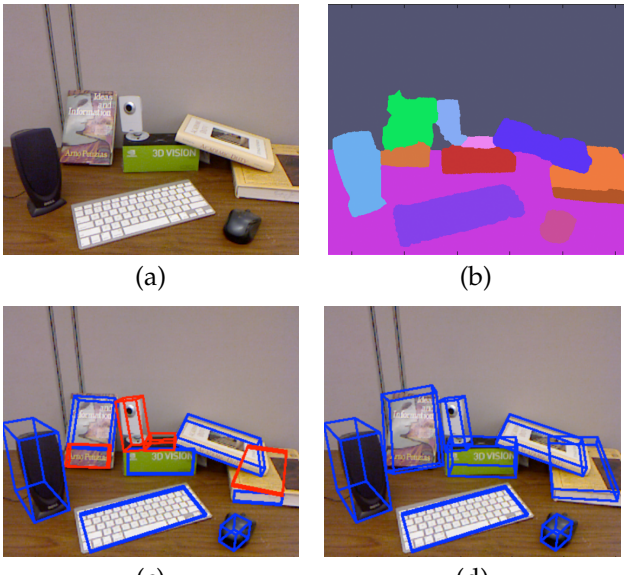


Fig. 1: (a) The input RGB-D image. (b) Initial segmentation from RGB-D data. (c) A 3D bounding box is fit to the 3D point clouds of each segment, and several features are extracted for reasoning about stability. Unstable boxes are labeled in red. (d) The segmentation is updated based on the stability analysis and 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., [5], [6], [7]), they are limited to single support or isolated objects on a flat

- *Zhaoyin Jia, Andrew Gallagher and Tsuhan Chen are with the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY 14850.*
  E-mail: {zj32, acg226, tc345}@cornell.edu
- *Ashutosh Saxena is with the Computer Science Department, Cornell University, Ithaca, NY 14853.*
  E-mail: asaxena@cs.cornell.edu

† *This work was first presented at Computer Vision and Pattern Recognition (CVPR), 2013 as an oral.*

surface. Although these methods work well on larger structures such as furniture and buildings, they do not apply to more complicated stacking arrangements of objects that can occur, for example, on desks or other cluttered situations.

In our algorithm, we first fit a 3D box to the point-cloud of each segment, and then extract several features for further reasoning 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 process whose parameters are learned from labeled training images.

The block representation provides us many useful features, such as the box fitting error and the object stability, and we learn the importance of each feature through supervised learning. We design an energy function to describe the quality of the segmentation given a RGB-D image (composed of a color image and its corresponding depth image). By minimizing this energy function value, we achieve a better scene segmentation and volumetric block representation. For minimization, we use a sampling algorithm that incorporates randomized moves including splitting and merging current segments.

We experiment on several datasets, from a synthetic block dataset to the NYU dataset of indoor scenes. We also propose a new Supporting Object Dataset (SOD) with various configurations and supporting relations, and a Grocery Dataset (GD) extended on SOD in order to demonstrate more application scenarios. Experimental results show that our algorithm improves RGB-D segmentation. Further, the algorithm produces 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 blocks.
2) The use of physics-based stability for modeling an RGB-D scene.
3) A learning-based framework for inferring object segmentation in an RGB-D scene.
4) New supporting objects datasets including segmentation labels and support information.

The rest of the paper is organized as follows: we discuss the related work in Section 2. An overview of the approach is presented in Section 3. We then present our approach for single box fitting in Section 4, and the features to model the pairwise box relations in Section 5. We present the stability reasoning process in Section 6. We introduce our energy function for segmentation in Section 7, including the sampling algorithm with splitting and merging. The experimental results are presented in Section 8. We conclude the paper and discuss future work in Section 9.

2 Related work

3D Understanding from Color Image: Object segmentation on a color image is one of the most studied computer vision problems, and many methods have been proposed, e.g., [8], [9], [10] These methods group pixels into objects by clues such as color, texture or semantic classification results. They operate on a 2D image, but it is a natural next step to incorporate 3D understanding into object segmentation.

The first attempts for geometric inference from a single color image were proposed in [1], [3] and [4] for estimating the depth of each segment using only color features. Usually, a ground plane is detected, and then the depth of a segment that stands on the ground can be estimated by the touching position. The results appear either as "pop-up images" [2]: segments stand like billboards in different depth layers and have empty space behind them, or as a "point-wise depth-grid" [1] or "piecewise planar segments" [4]: pixels or super-pixels are predicted as 2.5D depths. The limitation is obvious: these models do not align with our understanding of the scene, where each object actually occupies a *volume* in 3D, which we explore in this work (Fig. 1 (d)).

To overcome this limitation, Gupta et al. [5] propose a block-world representation to fit 2D color segments. This block world assumption has been proposed early in 1965 [11]. Following this assumption, Gupta et al. represent the segments in outdoor scenes by one of eight predefined box types representing a box viewed from various positions. Although buildings in these outdoor scenes may 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. [12] also use blocks representation for objects, but required interactive human labelings for non-box objects. Xiao et al. [13] detect 3D cuboids with arbitrary orientations solely in RGB images, Bleyer et al. [14] show box fitting for improved stereo, and Jiang et al. [15] propose a linear programming for fitting cuboids in depth images. 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 [16] [17] [18] [19]. Bao et. al in [20] combine object detection with supporting layer inference to achieve 3D scene understanding. Further, the 3D structure of indoor scenes has been studied through affordances, as in [21] [22] and [23]. Indoor images have the strong clues of lines and planes as well as a

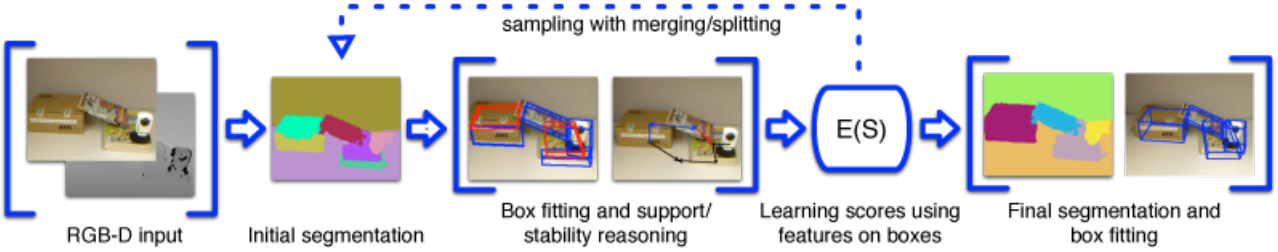


Fig. 2: An overview of our algorithm.

fixed composition of ceiling, wall and ground. These approaches posit that indoor spaces are designed by humans, so furniture items and objects are arranged in ways to facilitate usefulness of these spaces by humans. These approaches incorporate higher level understanding of between the scene and human interactions, and are complementary to ours.

RGB-D Scene Understanding: Previous work has shown that integrating depth with color information improves the performances of many vision tasks, such as segmentation (in [7]), contour detection (in [24]), object recognition (in [25], [26], and [27]), scene labeling (in [28], [29], [30] and [31]), and activity detection (in [32], [33] and [34]). These algorithms usually treat depth as another information channel without explicitly reasoning about the space that an object occupies. For example, when an object is partially observed from a single viewpoint, it remains hollow inside. In this way, segmentation and supporting inference are transformed into a classification problem in a 2.5D space, where the depth information of each visible pixel is available, but not the full 3D voxels of the objects that occupy the space. Koppula et al. [28], [30] considered 3D point-clouds but not fully volumetric reasoning. In contrast, we explicitly reason about full 3D models by fitting boxes to objects. This leads to a more natural interpretation of the scene, facilitated by better segmentation and support inference.

Support and Stability: Brand et.al. in [35] and [36] propose the vision system "SPROCKET" to analyze geared machines with basic knowledge of physics, such as friction and attachment. However their settings are constrained and difficult to be applied into general vision problems. Recently Grabner et.al. [37] analyze the interaction between humans and objects such as chairs in 3D space. The algorithm finds 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. [6] [38] 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 work, Silberman et al. [7] identify which image segments support which other segments. However, reasoning about support and stability are two different things. Past work on support presupposes 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.

In concurrent work, Zheng et al. [39] reason about stability in a depth image. They use geometric primitives, including voxels, to represent object volumes, and merge together neighboring voxels until stability is achieved. Their approach focuses only on the depth domain. In contrast, our work fuses both color and depth features. We model each object with cubic volumes and combine this representation with color information for reasoning about support, stability and segmentation in one framework.

We use a simple model for evaluating the stability of our block arrangements, although more complicated physics-based simulators [40] 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 the objects must project within its region of support. This simple model is justified by the ideas of intuitive physics [41] that humans even have 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 improving parsing 3D scenes.

3 Approach Overview

Our input is an initial RGB-D segmentation, generated from Silberman et al. [7]. First, we fit a 3D bounding box to the 3D point-cloud points corresponding to each segment. Next, we compute features for single boxes and between pairs of boxes and propose supporting relations, perform stability reasoning, and adjust the box orientation based on the supporting surfaces. Finally, we model the segmentation with an energy function based on learned regressors that are trained using these features. The segmentation is optimized by minimizing this energy function using

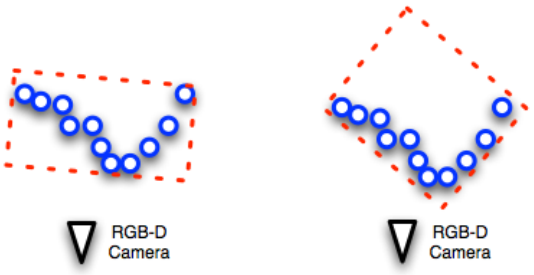


Fig. 3: (a) A bounding box fit based on minimum volume may not be a good representation for RGB-D images, where only partially observed 3D data is available. (b) A better fit box not only occupies a small volume, but also has many 3D points near the box surface. Data points are projected to 2D for illustration.

randomized splitting and merging. The output is the segmented RGB-D image along with volumetric representation using the fitted boxes and support information. See Fig. 2 for an overview.

4 Single Box Fitting

In this section, we describe the procedure for representing a segment from an RGB-D image with a box. RGB-D data is observed from only one viewpoint, and fitting 3D bounding boxes with minimum volumes [42] may fail to produce box representations that align well with the actual objects in the scene. Fig. 3 (a) 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. 3 (b).[1] We propose a RANSAC-based algorithm (details below) to fit boxes to the point cloud.

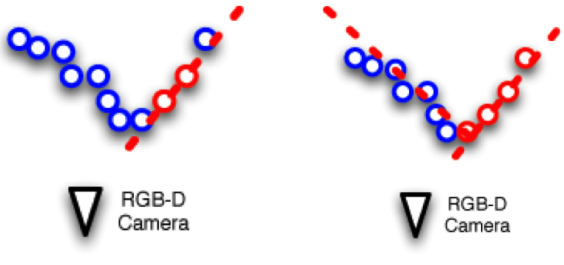


Fig. 4: (a) To fit the 3D points, we use RANSAC to find the first plane S_1. (3D points are projected on 2D for a simpler illustration, and the plane S_1 is presented as red line). (b) For the 3D points that do not belong to S_1, we fit another plane S_2 to them, enforcing that S_2 is perpendicular to S_1.

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 its nearest-face distance $\{D_{min}(P_i)\}$. The objective for our box fitting algorithm is to minimize this sum for all the 3D points: $\sum_i D_{min}(P_i)$.

The input to this step is the 3D points within one segment. First, we use RANSAC to find a plane to fit all the 3D points, providing the first surface S_1, shown in Fig. 4 (a). Next, we collect the outlier 3D points that do not belong to S_1, and then fit a plane, S_2, to them also using RANSAC. In experiments, we empirically set this threshold to 2cm, i.e. a point does not belong to a surface if its distance to this surface is larger than 2 cm. RANSAC for finding the normal is repeated for 10 times. We constrain that the surface orientation of S_2 is perpendicular to S_1, shown in Fig. 4 (b).

The above steps give the orientations that align 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 in front or behind, a large increase of the box volume can occur. Therefore, we allow for up to 5% outliers in the 3D points, requiring that $\geq 95\%$ of a segment's 3D points are enclosed within its box.

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 M times ($M = 15$ in our experiment setting) and the best fitting box (smallest distance $\sum_i D_{min}$) is chosen.

Algorithm 1 Minimum Surface Distance Box Fitting

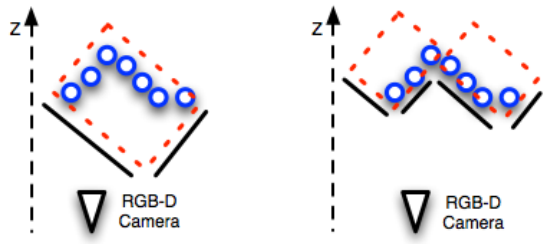
Given 3D Points $\{P_i\}$, M
Fit a plane S_1 to P, find outliers P_2 not belonging to S_1.
Fit a plane S_2 to P_2, such that S_2 is perpendicular to S_1.
Given S_1 and S_2, find the extend of the 3D box by minimum volume. Calculate $\sum_i D_{min}$.
Repeat M times to find the best fitting box.

4.2 Visibility

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

Fig. 5 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 points that belong to visible surfaces, and use this as the feature for later processing.

1. Recent related work [43] considered cylinder fitting of 3D points to the surface but also did not consider visibility.



(a) (b)

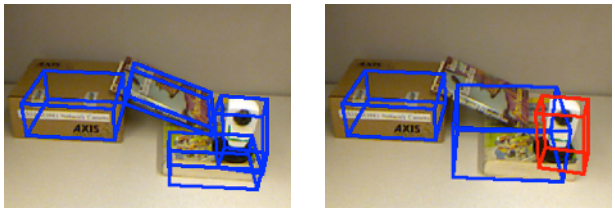
Fig. 5: Given the camera position and a proposed bounding box, we determine the visible surfaces of the box, shown as a solid parallel black line to the box surface. (a) This box may give a compact fit, but most of the points lie on the hidden surfaces. (b) With a better box fit, most of the points lie on the visible surfaces of the two boxes.

5 Pairwise Box Interaction

We examine two pairwise relations between nearby boxes: box intersection and box support. These features are important because they encode agreement between neighboring segments and provide additional clues for refining the box representation.

5.1 Box intersection

Box intersection gives an important clue for volume reasoning. Ideally, a box fitted 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. Fig. 6 shows an example.

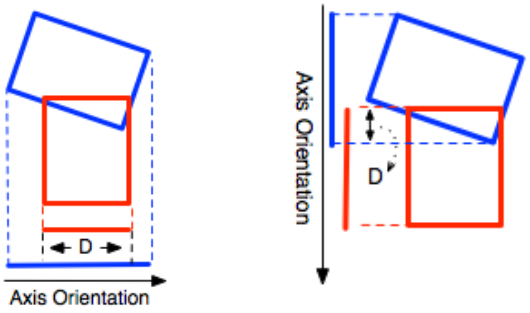


(a) (b)

Fig. 6: (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.

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) [44], used in computer graphics for collision detection. We present a 2D illustration for finding the distance of the box intersection in Fig. 7. The distance D shown in Fig. 7 (b) 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 to each surface, and project the vertexes of the two boxes to this plane. We compute the convex hull of the projection of each box, checking whether the two convex hulls intersect to find the minimum separating distance D.



(a) (b)

Fig. 7: Separating Axis Theorem in 2D: (a) in order to separate two boxes, we choose an axis perpendicular to any of the edges, 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 axiss (in this case four possibilities, two for each box), and choose the minimum overlap distance. This gives the orientation and the minimum distance required to separate two boxes.

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 object-object support scenarios, we define three supporting relations between the boxes: 1) surface on-top support (an 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. 8 (a) to Fig. 8 (c).

To classify supporting relations, we detect the ground and compute the ground orientation following [7]. 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 'side' support: an 'on-top' support has a separating direction nearly parallel to y-axis ($< 20°$), while a '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. 8(a), and a tilted on-top support, shown in Fig. 8(b). We distinguish these two types by examining the two closest surfaces of the pairwise boxes. If these two surfaces have a large angle difference ($> 20°$) between them, and have different orientations to the ground plane, then we classify it as a partial 'on-top' support, i.e., the object on top is tilted. Otherwise as a 'surface on-top' support.

Reasoning about stability requires that we compute centers of mass for object volumes, and determine

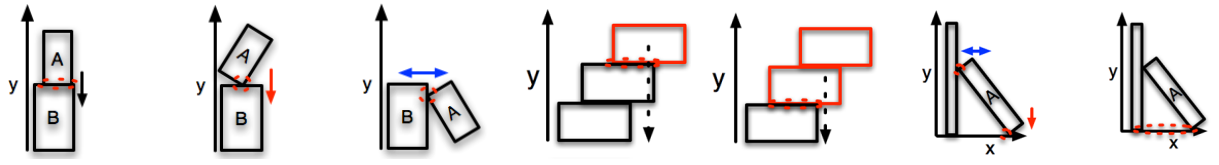


Fig. 8: (a) to (c): three different supporting relations: (a) surface on-top support (black arrow); (b) partial on-top support (red arrow); (c) side support (blue arrow). Different supporting relations give different supporting areas as plotted in red dashed circles. (d) to (e): stability reasoning: (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.

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. 8(a) to Fig. 8(c).

6 Global Stability

Box stability is a global property: boxes can appear to be supported locally, but still be in a globally unstable configuration. Fig. 8(d)-(e) provide an illustration.

We perform a top-down stability reasoning by iteratively examining the current gravity center and supporting areas. To determine the direction of the gravity, we first find the ground plane following the heuristics proposed in [7]: many surfaces in an indoor scene will follow the three orthogonal directions, and we choose the one that is closest to $Y = 1$ in the camera coordinates as the ground plane normal. The negative direction of the ground plane normal is used as the gravity direction.

The top-down stability process is shown in Fig. 8. For simplicity we assume each box has the same density. This assumption is usually valid for daily objects, e.g. books, boxes, or bottles. They have similar densities, and can either support other objects or be supported.

We begin with the top box by finding its 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, this time finding the center of mass of the set of boxes already found to be stable with the one under consideration. Assuming constant density, 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:

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

We iteratively update the center of mass by adding the boxes from top to bottom until the ground is reached. If we found that the current supporting area does not support the center of mass, we label the current box (or collection of boxes) unstable, shown in Fig. 8 (e). 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. 8 (f) to Fig. 8 (g).

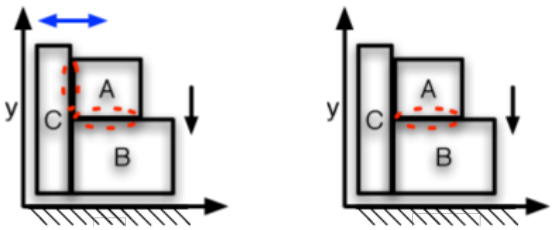
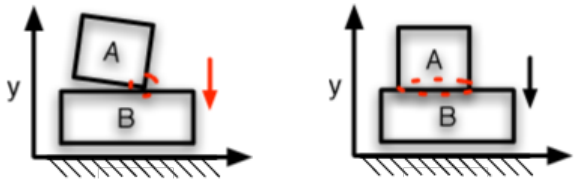


Fig. 9: (a) Near-touching objects, e.g., objects A and C do not necessarily support one another. (b) After stability reasoning, we find that object A can be fully supported by object B beneath it through a surface on-top support. Therefore, we delete the unnecessary side support between A and C.

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, we find a minimum combination of the on-top supports (both surface and partial) and at most two side supports examine whether the object can be stable. If so, all other support relations for the object are deleted. One example is shown in Fig. 9.

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 re-fit the 3D bounding box of the top object, confining the rotation of the first principle surface S_1 to be the same as the supporting surface. One example is illustrated in Fig. 10. We perform this



(a) (b)

Fig. 10: (a) 3D oriented bounding boxes can be ill-fit because of noise, and this may lead to incorrect support relation inference. For example, between object A and B, a partial on-top support is proposed, although it should have been a surface on-top support. (b) After stability reasoning, we adjust the higher box if it is only supported from beneath, and then correct the support relation accordingly.

adjustment on box-fitting every time after inferring the supporting relation and stability. This improves the 3D interpretation of the scene.

6.1 Integrating box-based features for segmentation

To incorporate all the box-based features, one baseline we implement is to start with an over-segmentation, and merge the pairwise segments based on learning (another possible implementation would be to start with an under-segmentation and perform splitting on each segment). We begin with initial segments generated with features from [7]. During training we use the ground-truth segmentation and 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. For example, to compute one type of features (surface distance) for a merge, 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 as the difference of this criterion before and after merging (1 dimension for each box before merging, 2 dimensions in total, denoted as **D**).

For this baseline model (labeled as **Stability** in the following sections), we train an SVM regression $y = w_{svm}^T x$ based on the 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 each newly merged segment, recompute the stability reasoning, and re-extract the features for regression. We repeat the above steps until the classifier does not classify any pair of segments as a pair that should be merged. Note that this baseline merges pairs of segments, has no backtracking, and must begin with an over-segmentation of the image.

7 A Learned Energy Function

In this section, we improve the baseline model (**Stability**) from the previous section by introducing an energy function with unary and pairwise terms based on the volumetric boxes, their support relations, and stability (this method is labeled as **R-Samp** for Randomized Sampling in the following sections). This model provides the framework for exploring the space of an energy function that represents the goodness-of-fit of a particular box representation and corresponding segmentation for a scene with the corresponding RGB-D input. We define two different moves, *splitting* to split a segment, and *merging* to merge two adjacent segments. These moves allow us to traverse the space over which the energy function is defined. We explore the space with a partical-based filter to discover a local minimum that, hopefully, corresponds to a good segmentation and box representation of the scene.

We use s_i to represent one individual segment in a segmentation, and denote a segmentation as $S = \{s_1, ..., s_N\}$ with N segments and M pairs of neighboring segments. We define a pool of segmentations as $\{S\}$, which includes a set of possible different segmentations given the RGB-D input. $\{S\}_{all}$ indicates the space of all possible segmentations. Given one particular segmentation S, 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 this segment, and it is learned using single box features including box fitting errors, volumes, and stability, described as x_i in the top of Table 1. Formally, $\phi(s_i)$ is defined as:

$$\phi(s_i) = w_s^T x_i, \qquad (3)$$

where w_s is the learned regression parameters.

Similarly, $\psi(s_i, s_j)$ is a regression score of two neighboring boxes. It is learned using pairwise box features including box intersection distance, pairwise support relations, and pairwise box features, x_{ij}, described in the bottom of Table 1. $\psi(s_i, s_j)$ is formally defined as:

$$\psi(s_i, s_j) = w_p^T x_{ij}, \qquad (4)$$

where w_p represents the regression parameters.

7.1 Single and pairwise potentials

In the following section we further explain the training and testing processes for our single and pairwise potentials that comprise our energy function. The input at this step is a mid-step segmentation S, including N segments and M pairs of neighboring segments. This initial segmentation can be generated using the algorithm proposed in the literature, e.g. [7], or the previously proposed algorithm **Stability**.

First, we learn the quality of each single segment s_i through a SVM regression as the single box potential $\phi(s_i)$. This is done through a supervised learning process on a held-out training set, and we generate the positive and negative training samples as follows:

TABLE 1: Features for single and pairwise potentials. The "relative" feature values are the features divided by the volume of the box, instead of the absolute value.

Single potential $\phi(s_i)$ features x_i	**dim**
Box orientation with respect to the ground	1
Mean and variance of the minimum surface distance	2
Mean and variance of the relative minimum surface distance (divided by box volume)	2
3D point density over volume	1
Percentage of the visible points	1
Number of intersecting boxes	1
Global Stability	1
Stabilities of the objects	1
Average (and relative) intersecting distance of the boxes	2
Distance (and relative distance) of the projected gravity center to the supporting area center	2
Distance (and relative distance) of the projected gravity center to the projected vertexes	16
Pairwise potential $\psi(s_i, s_j)$ features x_{ij}	**dim**
Number of intersection of each box	2
Relative of collision of each box (divided by each box volume)	2
Stability of each box	2
Pairwise supporting relations	1
Is one supporting another	1
Pairwise volume center distance	1
Projected gravity center to the supporting area center (if supported)	1
RGB-D features proposed in [7]	51

in the training images, we first use the ground-truth segmentation from human labeling as the positive training samples. We also make some random modifications from these ground-truth segmentations by splitting and merging, providing more positive and negative training instances. Then, we compute the segmentation score (the intersection-over-union ratio) of each segment s_i to the ground-truth segment $s_{j,gt}$:

$$score(s_i) = \max_{s_{j,gt}} \frac{Intersect(s_i, s_{j,gt})}{Union(s_i, s_{j,gt})}, \quad (5)$$

and consider a segment s_i as positive training sample if $score(s_i) \geq 90\%$, otherwise this segment is a negative training sample. After getting the training label, a 3D bounding box is then fit to this segment, and then the proposed box-related features x_i are computed for training.

During testing, we fit a 3D bounding box to each segment s_i, compute the features x_i, and perform the regression in Eq. 3 to calculate the single box potential value $\phi(s_i)$. Fig. 11 (c) presents one example of our single box potentials during testing. The boxes of the segments are color-coded in a way that the lower potential value $\phi(s_i)$ of segment s_i is, the more blue its corresponding box is. It shows that our proposed single box potential value captures the segment quality and classifies the ill-fit boxes, e.g., the boxes with yellow and red colors.

The pairwise potential is trained and tested following the similar manner: multiple randomly generated segmentations as well as the ground-truth ones are processed during training. A boundary is considered

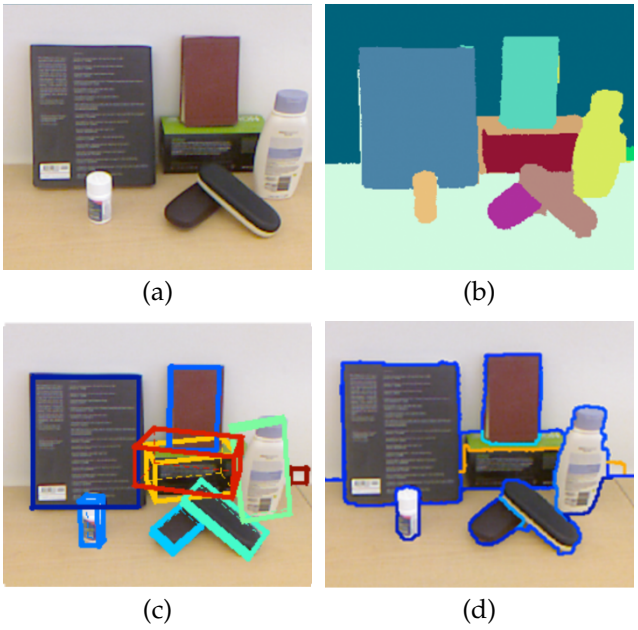


Fig. 11: (a) Input image. (b) Mid-step segmentation during testing. (c) and (d) are exemplar testing results for (c) single potential $\phi(s_i)$ and (d) pairwise potential $\psi(s_i, s_j)$. The color of the boxes and boundaries is coded as the better quality the segments are, the more blue the boxes and boundaries are, with lower potential values. Our proposed features capture the quality of each segment and boundary.

a positive training instance if the two segments it lies between both have segmentation scores (proposed in Eq. 5) larger than 90%. During testing, 3D bounding boxes are also first fit to all the segments, and then the pairwise features described in Table 1, bottom part, are computed. We perform regression $\psi(s_i, s_j)$ on the pairs of the segments sharing a boundary. Fig. 11(d) presents one example of pairwise potentials $\psi(s_i, s_j)$. This potential gives a good indication of which pairs of segments, if merged, might produce a reduction to the global energy function.

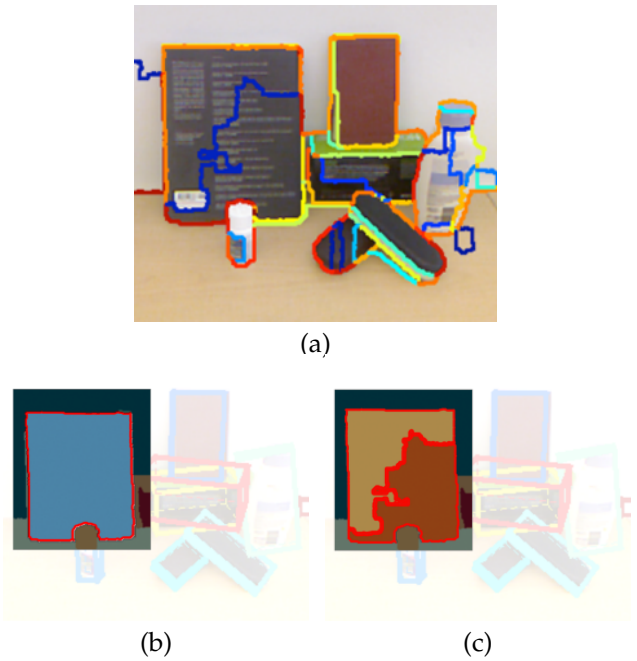
7.2 Minimizing through splitting and merging

During testing, our goal is to minimize this energy function and find the optimal segmentation S^* that has the minimum energy value:

$$S^* = \arg\min_S E(S). \quad (6)$$

Note that this energy function is non-convex, and the space of possible segmentations $\{S\}_{all}$ is very large, therefore it is infeasible to perform an exhaustive search to find the global minimum.

To explore the space, we adopt a Randomized Sampling (R-Samp) [45] 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) splitting one segment into two smaller segments, or (b) merging two neighboring segments into one segment. We use the potentials $\phi(s_i)$ and $\psi(s_i, s_j)$ to indicate which segments should be split or merged while designing



(a) (b) (c)

Fig. 12: (a) We pre-compute all the possible boundaries given RGB-D image. (b) The selected segment before splitting. (c) The selected segment after splitting. The splitting move is constrained to split one segment into two.

the sampling moves. We keep a pool of possible segmentations as the particles to explore this energy space, and keep track of the ones with the minimum energy values as we iterate for optimization.

Splitting: The single box potential $\phi(s_i)$ indicates the quality of each individual segment. This value guides the splitting moves so that we explore the segmentation space in a more efficient manner.

We calculate the single box potential for all the segments in the current segmentation, and then randomly choose one segment s_i to split based on its potential $\phi(s_i)$: the higher $\phi(s_i)$ is, the more likely s_i is going to be selected for splitting, because it represents a worse segmentation quality for s_i, and thus s_i needs to be modified. The final likelihood of selecting one segments is linearly mapped from $\phi(s_i)$ by converting $\phi(s_i)$ into probabilistic prediction [46].

Specifically, we split one segment s_i as follows: we pre-compute a boundary map of all the possible edges given the RGB-D images using [7]. One example is shown in Fig. 12(a): all the possible boundaries are presented in this boundary map, including the false ones. This map provides us the basis for splitting one segment. Then given the selected segment s_i, this segment is forced to be split into two segments based on the boundary map, as illustrated in Fig. 12(b) and Fig. 12(c). The boundaries within s_i are merged from lower values to higher values based on the pre-computed boundary map, until only two segments remain in s_i.

Merging: We merge the segments with a similar principle: first we compute all the pairwise potentials $\psi(s_i, s_j)$ given the current segmentation, and then we randomly sample a pair of segments based on their pairwise potential value: if two segments s_i and s_j have a higher pairwise potential $\psi(s_i, s_j)$, they have a higher chance to be selected for merging, because $\psi(s_i, s_j)$ indicates a worse quality boundary between two segments. After the boundary and its pair of segments are chosen, we merge the neighboring two segments by deleting the boundary between them and group all the pixels into one segment.

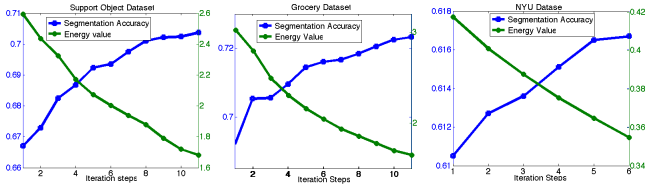
Minimization: The energy function in Eq. 2 is devised in the way that the smaller the value is, the better segmentation is. We find a better segmentation with a lower energy value by maintaining a segmentation pool $\{S\}$, and repeatedly finding the segmentations with smaller energy values within this pool. Splitting and merging compose our basic moves for minimization. Given one initial segmentation, we propose $2N$ (we use $N = 5$) new segmentations by N splitting moves and N merging moves, and then re-evaluate all segmentations using Eq. 2. We take the K (we use $K = 5$) segmentations with the smallest energy values for the next iteration, and discard the remaining segmentations. We repeat this step again, so that the top K segmentations will branch, producing KN new moves, and then be evaluated together to choose the top K segmentations for the next step. We repeat this sampling step until we reach the maximum number of iterations M. In practice, this algorithm optimizes our energy function to a reasonable local minima in about 10-15 iterations. The details of the algorithm are presented in Alg. 2.

Algorithm 2 Energy Minimization

Given constants N, K, and M.
Initialize segmentation pool $\{S\}$ with initial segmentation S_{init}.
for each segmentation S_t in the pool $\{S\}$ **do**
  Compute $\phi(s_i)$ and $\psi(s_i, s_j)$ for S_t.
  Sample one segment s_i by $\phi(s_i)$ and split it, producing new segmentation S_j
  Add S_j to $\{S\}$, repeat N times sampling of one segment.
  Sample one pair of segments by $\psi(s_i, s_j)$ and merge them, producing new segmentation S_j
  Add S_j to $\{S\}$, repeat N times sampling of a segment pair,
end for
Evaluate the energy function E for all the segmentations in $\{S\}$.
Keep top K segmentations in $\{S\}$ with smallest $E(S)$.
repeat M times
Output S^*_{final} with the minimum energy value $E(S)$ in the $\{S\}$.

Experimental Illustration: Now we illustrate that our energy minimization approach improves performance over iteration steps in Fig. 13, for the three different datasets that we experiment on (see next section). The overall average segmentation performance is shown as blue curve and the average energy function value

Fig. 14: The segmentation results improve along with more iterations of the proposed algorithm **R-Samp**. Given the color image (a), and the depth image (b), the initial segmentation (c) may have some mistakes. Some of these mistakes are corrected during middle steps as iteration goes on, shown in (d). In the final iteration, the segmentations are corrected into more reasonable ones, presented in (e).



(a) (b) (c)

Fig. 13: Segmentation results of our proposed sampling algorithm (**R-Samp**) over each iteration on the SOD dataset (a), GD dataset (b) and NYU-2 dataset (c). As the energy value decreases through the minimization steps, the accuracy of the segmentation increases.

is shown as green curve. It shows that the accuracy of the segmentation increases as we minimize the energy function through our **R-Samp** sampling process. Therefore, it provides evidence that our energy function accurately represents the quality of the segmentation.

Fig. 14 presents a particular sequence of the top segmentations (smallest energy values) at each step as we minimize the energy function. Our moves of splitting and merging improve the overall segmentation.

8 Experiments

We perform experiments on four different types of datasets: a block dataset, a supporting object dataset (SOD), a grocery dataset, and a public dataset of indoor scenes proposed in [7]. We evaluate the box fitting accuracy, the support relation prediction, and the segmentation performance.

8.1 Datasets

First we describe the datasets we have applied for evaluating our proposed algorithms.

Block dataset: We apply our algorithm to a toy block dataset. This dataset has 50 RGB-D images of blocks (see Fig. 15 and 16). For each block, we manually provide the ground-truth segment labels, as well as the orientations of two perpendicular surfaces[2]. 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 manually labeled.

2. The third surface orientation is perpendicular to the first two, and thus determined after providing the first two surface orientations.

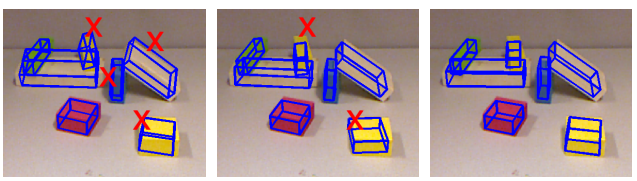
TABLE 2: Average angle error on the bounding box orientation given ground-truth segmentations.

	Block Dataset
Min-vol	15.41°
Min-surf	9.75°
Supp-surf	7.02°

Supporting object dataset: Many of the daily objects can be approximated as 3D volumetric blocks with similar densities, following our stability reasoning assumption. Thus 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 objects supporting it. (See Fig. 19.)

Grocery dataset: One possible application scenario of our proposed algorithm is a supermarket, where many objects are contained in regular boxes. We collect an extended Grocery Object dataset (GD) based on the Support Object Dataset (SOD) to demonstrate this application. This dataset mimics the environment of a grocery store, and includes a variety of common grocery objects, such as cereal boxes, shampoo bottles, etc. The dataset contains 609 RGB-D images with human-labeled ground-truth segmentation. (See Fig. 20.)

NYU indoor dataset: Finally we evaluate segmentation performance on the newly released RGB-D NYU-2 indoor dataset [7].



(a) (b) (c)

Fig. 15: Fitting results on the block dataset. (a): **Min-vol**. (b): **Min-surf**. (c): **Supp-surf**. Blocks with large fitting error in orientation are labeled as a red "x".

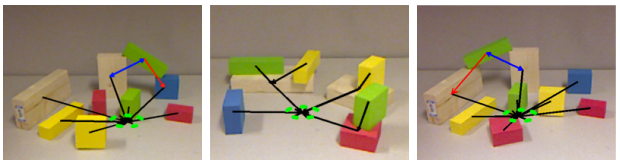


Fig. 16: The predicted supporting relations on block dataset. Three different types of the supporting relations are colored in black (surface-top), red (partial-top), and blue (side). The ground plane center is plot as a green dashed circle.

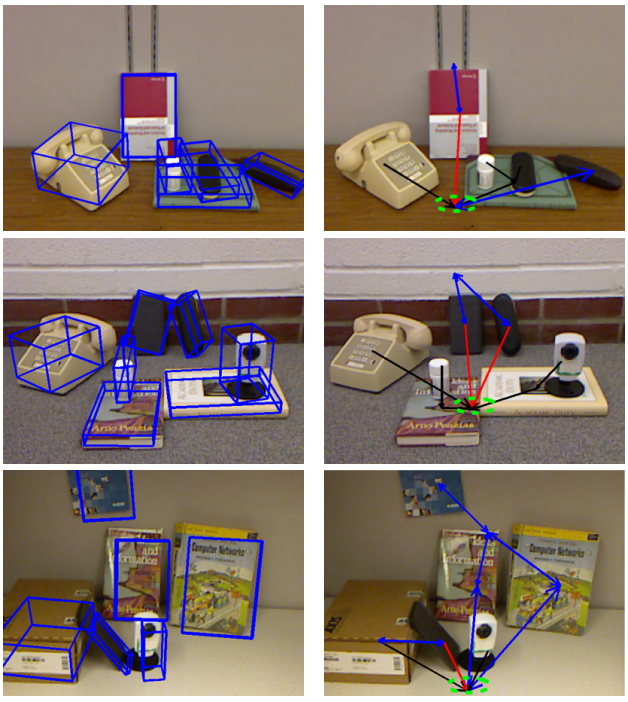


Fig. 17: We qualitatively show our box fitting algorithm (left) on daily objects with ground-truth image segmentation and the supporting relation prediction after stability reasoning (right). Boxes for large surfaces (like the back wall and the ground) are not displayed for better visualization. The ground plane is plotted as a green dashed circle for showing the support inference results.

Fig. 18: 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.

8.2 Box fitting and support relation prediction

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

Min-vol: the baseline algorithm from [42] 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 3 shows that our proposed minimum surface distance provides a better box fitting compared to the minimum volume criteria, reducing the errors in angle from $15.41°$ to $7.02°$, a 40% improvement. With stability reasoning, the fitting decreases error by another $2°$ in absolute value, a 15% improvement.

We then analyze the performance of our stability reasoning. We compare with the ground truth supporting relations, and count an object as correct if all its supporting objects are predicted. We compare our proposed algorithm (**Stability Reason**) 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: Supporting relation accuracy for different datasets.

	Block	SOD
Neighbor	80.59%	52.88%
Stability Reason	**91.68%**	**72.86%**

TABLE 4: Pixel-wise segmentation score.

	SOD	GD	NYU
[7]	60.2%	65.9%	60.1%
S/P	64.7%	68.1%	60.8%
Stability	66.7%	69.2%	61.0%
R-Samp	**70.0%**	**72.3%**	**61.7%**

Table 3, left column reports the supporting relation accuracy for this block dataset. 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. 16.

We measure the prediction of the supporting relations with the ground truth segmentation for support relation dataset. The results of using the baseline **Neighbors** and our stability reasoning **Stability Reason** are shown in Table. 3, right column. 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. 17 presents the exemplar results of our box fitting and support prediction from the supporting object dataset.

8.3 Segmentation evaluation

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 [7] 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 randomized sampling allowing both merging and splitting (**R-Samp**). The segmentation accuracy is scored by pixel-wise overlapping with the ground-truth segments, proposed in [3] and [7].

Supporting Object Dataset: Table 4, first column, shows the performance comparison with different fea-

ture sets for our proposed Supporting Object 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 randomized sampling gives the best results with another 3% improvement. Testing results with block fitting are shown in Fig. 19. In general, the final algorithm (**R-Samp**) performs better as we further iterate the sampling steps.

Grocery Dataset: we also evaluate the segmentation accuracy on the Grocery Dataset, and compared it with the baseline algorithm proposed in [7]. Full quantitative results of different algorithms are presented in Table 4, middle column. The results show that our proposed new feature set increases the segmentation accuracy, and the final sampling algorithm (**R-Samp**) with merging and splitting moves gives the best result. Some example testing images with final block representations are presented in Fig. 20. Our final algorithm produces more reasonable segmentation results as well as the volumetric block representations. This provides a richer interpretation of the object in the scene.

NYU-2 indoor dataset: Finally we evaluate segmentation performance on the newly released RGB-D NYU-2 indoor dataset [7], and report the performance in Table 4, right column. This dataset is proposed for scene understanding, rather than object reasoning, and many large surfaces, such as counters and drawers, and are sometimes labeled as two or more distinct objects, i.e., one for each surface, instead of one for the entire object. Although these conditions limit the evaluated performance of our proposed algorithm, adding the proposed features still improves the segmentation performance. The performance of our sampling algorithm (**R-Samp**) gives the best results, also the performance improves throughout iteration steps. (See Fig. 13.)

We find that although proposed for modeling small object interactions, this block representation and stability reasoning framework can also be extended to some indoor scenarios, e.g., for furniture sitting on the ground or supported on the wall. We qualitatively present the box fitting and supporting inference result with ground-truth segmentation for a indoor bedroom scenario in Fig. 18.

Some exemplar segmentation testing results are presented in Fig. 19 and Fig. 20.

Speed: Our algorithm is implemented in Matlab, and the processing time for one RGB-D image takes three to five minutes, including watershed initialization, extracting g-Pb boundaries, color and depth features, and bounding box fitting. Overall the processing time is in about the same degree in comparison to the literature [7].

9 Conclusions and Future Work

In this paper, we propose analyzing RGB-D images through physically-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 subsequently modified for consideration in later iterations. Stability reasoning produces better estimates of supporting relations (by requiring enough support to provide stability for each object) and improved box orientation estimates (by knowing when objects are fully or partially supported from below). Experiments show that our proposed algorithm works for both synthetic and real world scenes, and leads to improvements in box fitting, support detection, and segmentation.

We believe that physics-based stability reasoning in segmentation could be useful in several applications with RGB-D data, for example, activity detection, object detection and tracking, scene modeling, applications to robotics, and so on. We mention a few possible future directions that can be extended based on the algorithm proposed in this paper.

3D oriented block fitting with color image: The block fitting algorithm in this paper solely relies on the 3D point clouds. However as presented in contemporary work [13], color channel provides informative edge clues, which can be incorporated together with point-clouds for improving the bounding box fitting.

Extending primitive shapes: Although blocks are good approximations for many convex objects, there are cases when they limit the performance of scene reasoning. For example, a basketball may be failed to be presented as a 3D oriented bounding box, and therefore its stability cannot be correctly estimated using the simple blocks that we propose. Extending the primitive shapes from blocks to cylinders (e.g., [43]) spheres, or non-parametric shapes, along with corresponding advancements to the stability reasoning module, may improve the support and stability reasoning, as well as the final object segmentation.

Combining with semantic classification: Previous work has shown that combining different tasks improves performance of individual vision tasks [47], [48]. We believe that combining the block representation with semantic classification will further improve the 3D scene understanding. Concave objects, such as chairs, are not well represented by a single box. In these situations, we can use multiple boxes to build the objects. Scemantic classification can also be performed on other attributes, e.g. estimate the block density. To prevent the system from over-fitting, we could detect objects as a pre-processing step, and then propose potential category hypothesis for the

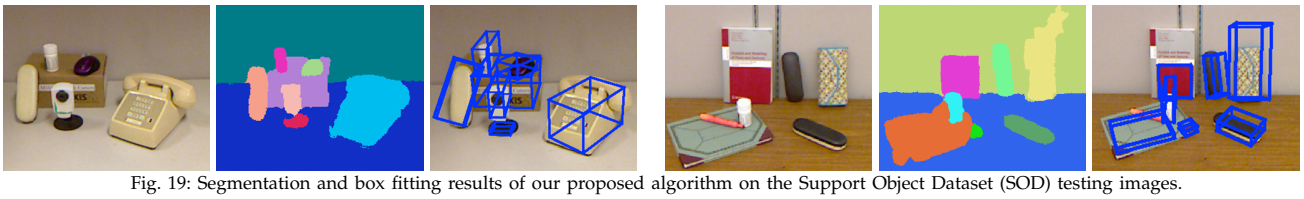


Fig. 19: Segmentation and box fitting results of our proposed algorithm on the Support Object Dataset (SOD) testing images.

Fig. 20: Segmentation and box fitting results of our proposed algorithm on the Grocery Dataset (GD) testing images.

target objects. After that, we can choose the correct number of blocks to approximate the object better.

Hidden support: In this work we assume all the support relations are visible in the scene. However it is possible to analyze the hidden supports that are invisible or occluded. There are possible clues that enable us to do this: for example, if one box is tilted with no other neighboring support, it is likely that the object is supported by an invisible object, e.g., a glass, or the supporting object is completely occluded. Analyzing the hidden support will unify the stability reasoning with the concept of occlusion.

Completing the physical model: For reasoning about stability, our model makes broad assumptions about objects in the scene. We assume objects are constant density and that objects are supported when their center of gravity projects into the convex hull of support, effectively ignoring friction. Further, we only reason about stability in a top-to-bottom fashion. Other, more sophisticated physical modelers (e.g., [40], Bullet [49] or Open Dynamics Engine [50]), though computationally more expensive, could be also used. We expect they would provide a more complete analysis of the physics and lead to better RGB-D segmentations.

Applications to Robotics: For a robot performing manipulation tasks, perceiving such a physically correct interpretation of a scene is very useful. In this paper, we have tested our approach on the grocery dataset. In future work, a robot planning algorithm (e.g., [51]) could use our segmented output for making a grocery checkout robot.

ACKNOWLEDGEMENT

We thank Daniel Jeng for useful discussions about stability reasoning. This work is supported in part by NSF DMS-0808864 and Qualcomm, and by NSF Career award (to Saxena).

REFERENCES

[1] A. Saxena, S. H. Chung, and A. Y. Ng, "Learning depth from single monocular images," in *NIPS*, 2005.

[2] H. D, A. A. Efros, and M. Hebert, "Recovering surface layout from an image," *IJCV*, vol. 75, no. 1, pp. 151–172, 2007.

[3] D. Hoiem, A. N. Stein, A. A. Efros, and M. Hebert, "Recovering occlusion boundaries from a single image," in *ICCV*, 2007.

[4] A. Saxena, M. Sun, and A. Y. Ng, "Make3D: Learning 3D scene structure from a single still image," *PAMI*, vol. 31, no. 5, 2009.

[5] A. Gupta, A. A. Efros, and M. Hebert, "Blocks world revisited: Image understanding using qualitative geometry and mechanics," in *ECCV*, 2010.

[6] Y. Jiang, M. Lim, C. Zheng, and A. Saxena, "Learning to place new objects in a scene," *IJRR*, vol. 31, no. 9, 2012.

[7] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *ECCV*, 2012.

[8] P. Arbelaez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik, "Semantic segmentation using regions and parts," in *CVPR*, 2012.

[9] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *PAMI*, 2011.

[10] S. Maji, N. Vishnoi, and J. Malik, "Biased normalized cuts," in *CVPR*, 2011.

[11] L. G. Roberts, "Machine perception of 3-D solids," in *Optical and Electro-Optical Info. Proc.*, 1965, pp. 159–197.

[12] 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*, vol. 31, no. 4, p. 99, 2012.

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

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

[15] H. Jiang and J. Xiao, "A linear approach to matching cuboids in rgbd images," in *CVPR*, 2013.

[16] E. Delage, H. Lee, and A. Y. Ng, "A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image," in *CVPR*, 2006.

[17] A. Flint, D. W. Murray, and I. Reid, "Manhattan scene understanding using monocular, stereo, and 3D features," in *ICCV*, 2011.

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

[19] 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.

[20] S. Bao, M. Sun, and S. Savarese, "Toward coherent object detection and scene layout understanding," *Image and Vision Computing*, vol. 29, no. 9, 2011.

[21] D. F. Fouhey, V. Delaitre, A. Gupta, A. A. Efros, I. Laptev, and J. Sivic, "People watching: Human actions as a cue for single view geometry," in *ECCV (5)*, 2012.

[22] Y. Jiang, M. Lim, and A. Saxena, "Learning object arrangements in 3d scenes using human context," in *ICML*, 2012.

[23] A. Gupta, S. Satkin, A. A. Efros, and M. Hebert, "From 3D scene geometry to human workspace," in *CVPR*, 2011.

[24] X. Ren and L. Bo, "Discriminatively trained sparse code gradients for contour detection," in *NIPS*, 2012.

[25] 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.

[26] Y. Jiang, H. Koppula, and A. Saxena, "Hallucinated humans as the hidden context for labeling 3d scenes," in *CVPR*, 2013.
[27] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view RGB-D object dataset," in *ICRA*, 2011.
[28] H. Koppula, A. Anand, T. Joachims, and A. Saxena, "Semantic labeling of 3D point clouds for indoor scenes," in *NIPS*, 2011.
[29] N. Silberman and R. Fergus, "Indoor scene segmentation using a structured light sensor," in *ICCV Workshops*, 2011.
[30] A. Anand, H. Koppula, T. Joachims, and A. Saxena, "Contextually guided semantic labeling and search for 3d point clouds," *IJRR*, 2012.
[31] X. Ren, L. Bo, and D. Fox, "RGB-(D) scene labeling: Features and algorithms," in *CVPR*, 2012.
[32] H. S. Koppula and A. Saxena, "Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation," in *ICML*, 2013.
[33] H. Koppula, R. Gupta, and A. Saxena, "Learning human activities and object affordances from rgb-d videos," *IJRR*, 2013.
[34] H. Koppula and A. Saxena, "Anticipating human activities using object affordances for reactive robotic response," in *RSS*, 2013.
[35] M. E. Brand, P. R. Cooper, and L. A. Birnbaum, "Seeing physics or, physics is for prediction," in *Physics Based Modeling Workshop in Computer Vision*, 1995.
[36] M. Brand, "Physics-based visual understanding," *Computer Vision and Image Understanding*, vol. 65, no. 2, 1997.
[37] H. Grabner, J. Gall, and L. J. V. Gool, "What makes a chair a chair?" in *CVPR*, 2011.
[38] Y. Jiang and A. Saxena, "Infinite latent conditional random fields for modeling environments through humans," in *RSS*, 2013.
[39] B. Zheng, Y. Zhaoy, J. C. Yuy, K. Ikeuchi, and S. Zhu, "Beyond point clouds: Scene understanding by reasoning geometry and physics," in *CVPR*, 2013.
[40] D. Baraff, "Physically based modeling: Rigid body simulation," Pixar Animation Studios, Tech. Rep., 2001.
[41] M. McCloskey, "Intuitive physics," *Scientific American*, vol. 248, no. 4, pp. 114–122, 1983.
[42] C. Chang, B. Gorissen, and S. Melchior, "Fast oriented bounding box optimization on the rotation group $SO(3, R)$," *ACM Transactions on Graphics*, vol. 30, no. 5, 2011.
[43] D. Ly, A. Saxena, and H. Lipson, "Co-evolutionary predictors for kinematic pose inference from rgbd images," in *GECCO*, 2012.
[44] S. Gottschalk, "Separating axis theorem," *Tech Report*, 1996.
[45] J. Chang and J. W. Fisher, "Efficient MCMC sampling with implicit shape representations," in *CVPR*, 2011.
[46] R. C. W. C. Lin, "Simple probabilistic predictions for support vector regression," in *Tech Report*, 2004.
[47] L.-J. Li, R. Socher, and L. Fei-Fei, "Towards total scene understanding:classification, annotation and segmentation in an automatic framework," in *Computer Vision and Pattern Recognition (CVPR)*, 2009.
[48] C. Li, A. Kowdle, A. Saxena, and T. Chen, "Towards holistic scene understanding: Feedback enabled cascaded classification models," *PAMI*, vol. 34, no. 7, pp. 1394–1408, 2012.
[49] "http://bulletphysics.org," in *Bullet*.
[50] "http://www.ode.org/," in *Open Dynamics Engine*.
[51] A. Jain, B. Wojcik, T. Joachims, and A. Saxena, "Learning trajectory preferences for manipulators via iterative improvement," in *NIPS*, 2013.

Zhaoyin Jia is a Software Engineer with Google X, working on self-driving car project. He earned the PhD degree in Electrical and Computer Engineering from Cornell University in 2013. His thesis is focused on fusing depth information with RGB color image for a better scene understanding. He was a research intern in Eastman Kodak, 2011 and in Facebook, 2012, converting computer vision algorithm into applications. His research interest includes RGB-D image understanding, 3D reconstruction, robotics, egocentric vision and mobile vision.

Andrew Gallagher is a Senior Software Engineer with Google, working with geo-referenced imagery. Previously, he was a research scientist at Cornell University's School of Electrical & Computer Engineering, and part of a computer vision start-up, TaggPic, that recognized precise camera geo-locations from images. He earned his Ph.D. in ECE from Carnegie Mellon University in 2009. He was a research scientist for the Eastman Kodak Company until 2012, developing computational photography and computer vision algorithms for digital photofinishing, such as dynamic range compression, red-eye correction and face recognition. Andy is interested in a wide variety of data analysis problems, and has developed algorithms for detecting image forgeries, assembling jigsaw puzzles, recognizing people and social relationships in images, and deciding what NFL teams should do on fourth down.

Ashutosh Saxena is an assistant professor in the Computer Science department at Cornell University. His research interests include machine learning, robotics and computer vision. He received his Ph.D. in 2009 from Stanford University, and his B.Tech. in 2004 from IIT Kanpur, India. He has won best paper awards in 3DRR, RSS and IEEE ACE. He was named a co-chair of IEEE technical committee on robot learning. He has also received Sloan Fellowship in 2011, Google Faculty Award in 2011, Microsoft Faculty Fellowship in 2012, and a NSF Career award in 2013.

Ashutosh has developed robots that perform household chores such as unload items from a dishwasher, arrange a disorganized house, checkout groceries, etc. He has developed machine learning algorithms for perceiving environments from RGB-D sensors such as scene understanding, activity detection and anticipation. Previously, Ashutosh has developed Make3D (http://make3d.cs.cornell.edu), an algorithm that converts a single photograph into a 3D model. His work has received substantial amount of attention in popular press, including the front-page of New York Times, BBC, ABC, Discovery, FOX News, and Wired Magazine.

Tsuhan Chen has been with the School of Electrical and Computer Engineering at Cornell University since 2009, where he is the Director of the School, and the David E. Burr Professor of Engineering. From 1997 to 2008, he was with the Department of Electrical and Computer Engineering at Carnegie Mellon University, as Professor and Associate Department Head. From 1993 to 1997, he worked at AT-T Bell Laboratories, New Jersey. He received his M.S. and Ph.D. in electrical engineering from the California Institute of Technology, in 1990 and 1993, respectively. He received his B.S. in electrical engineering from the National Taiwan University in 1987.

Tsuhan served as the Editor-in-Chief for IEEE Trans. on Multimedia in 2002-04. He also served on the Editorial Board of IEEE Signal Proc. Magazine, and as Associate Editor for IEEE Trans. on Circuits and Systems for Video Technology, IEEE Trans. on Image Processing, IEEE Trans. on Signal Processing, and IEEE Trans. on Multimedia. He co-edited a book titled Multimedia Systems, Standards, and Networks. Tsuhan received the Charles Wilts Prize at the California Institute of Technology in 1993. He was a recipient of the NSF CAREER Award in 2000. He received the Benjamin Richard Teare Teaching Award in 2006, and the Eta Kappa Nu Award for Outstanding Faculty Teaching in 2007. He was elected to the Board of Governors, IEEE Signal Proc. Society, 2007-09, and a Distinguished Lecturer, IEEE Signal Proc. Society, 2007-08. He was elected as the Vice President of ECE Department Head Association in 2012, and currently serves as the President. He is a member of the Phi Tau Phi Scholastic Honor Society, and Fellow of IEEE.