Finding Approximate Convex Shapes in RGBD Images

Hao Jiang

Computer Science Department, Boston College, USA

Abstract. We propose a novel method to find approximate convex 3D shapes from single RGBD images. Convex shapes are more general than cuboids, cylinders, cones and spheres. Many real-world objects are near-convex and every non-convex object can be represented using convex parts. By finding approximate convex shapes in RGBD images, we extract important structures of a scene. From a large set of candidates generated from over-segmented superpixels we globally optimize the selection of these candidates so that they are mostly convex, have small intersection, have a small number and mostly cover the scene. The optimization is formulated as a two-stage linear optimization and efficiently solved using a branch and bound method which is guaranteed to give the global optimal solution. Our experiments on thousands of RGBD images show that our method is fast, robust against clutter and is more accurate than competing methods.

1 Introduction

Finding 3D structures from RGBD images is an important computer vision task. Instead of finding shape primitives such as planes, cuboids, spheres or cylinders, we extract shapes that are roughly convex from a single RGBD image. A RGBD image provides the information of a scene surface from a specific view. Each point on the surface has not only x, y and z coordinates but also a RGB color. The motivation to extract approximate convex structures in RGBD images is that most objects in both indoor and outdoor environments are roughly convex and all complex objects can be modeled as the composition of a set of convex shapes. By finding approximate convex shapes from RGBD images, we extract important structures of a scene. Fig. 1 illustrates finding approximate convex shapes in single RGBD images using the proposed method. As shown in the figure, we find a set of near-convex shapes, each corresponding to a set of 3D points in the point cloud of a RGBD image.

Finding approximate convex shapes in RGBD images is different from approximate convex decomposition [5,10] of mesh graphics models. Previous convex decomposition methods have been mainly used to simplify complex 3D graphics objects into approximate convex parts for easier collision control in animation. These convex decomposition algorithms usually require the input 3D mesh models to be noise free and represent a complete object scan, which has no inter- or intra-object occlusion and reveals the look of the object from each different

D. Fleet et al. (Eds.): ECCV 2014, Part III, LNCS 8691, pp. 582–596, 2014.

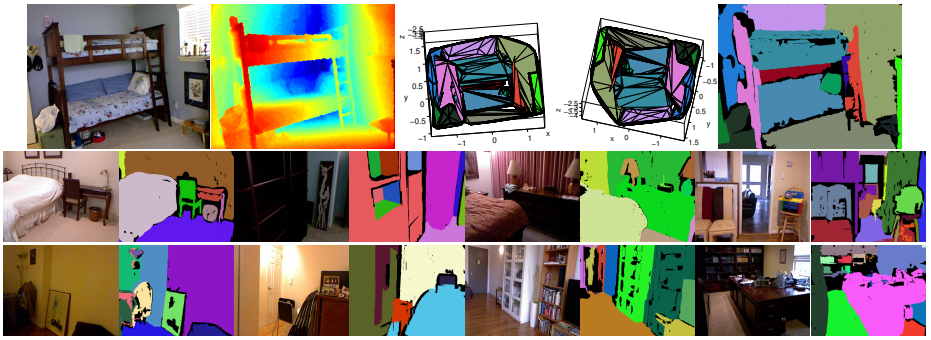


Fig. 1. Finding approximate convex shapes in single RGBD images. Row one shows a color image, its depth map, the convex hulls of the found shapes in two different views and the masks of the shapes on the image plane from the proposed method. Rows 2-3 show more example results of the proposed method (each result is a pair of color image and the approximate convex shape map).

view point. In this paper, we tackle the problem of finding approximate convex structures from a single view of the scene. There are usually heavy occlusions and data is missing in different places. The 3D point cloud is also quite noisy. This makes the problem of finding convex shapes in RGBD images challenging. Another difference of our shape finding task to previous pure geometric data decomposition is that apart from geometry structure we also have color information which helps disambiguate some object level boundaries. Moreover, instead of decomposing a 3D point cloud into roughly convex parts, our task is to find the approximate convex shapes and these shapes are not necessarily completely disjoint. This enables us to build a more efficient method that directly works on shape selection and is able to use shape proposals from different methods. To our knowledge, finding roughly convex shapes in RGBD images has not been attempted before.

In this paper we propose a global optimization method to find the approximate convex shapes in RGBD images by using surface patch candidates (3D superpixels). These candidates can be obtained from different methods by using both color and 3D shape information. The optimization selects shapes from the large set of candidates so that the overall convex fitting error is small, shapes have small overlaps, there is a small number of shapes and the extracted shapes mostly cover the RGBD image. We formulate the optimization problem as a two-stage mixed integer linear program. We find the globally optimal solution using a branch and bound method. Experiments show that our proposed method is more accurate than competing methods and at the same time is more efficient.

1.1 Related Methods

Finding geometric primitives such as planes, spheres, cones, cylinders [1,2] in a point cloud has been intensively studied. Representing objects in RGBD or stereo

images using cuboids has been studied in [12,17,18]. Methods that use geometric primitives tend to represent a scene with many small components if the scene contains objects with complex shapes. In [3], a greedy method is proposed to find superquadratics in clutter-free range data. In this paper, we detect approximate convex shapes, which provide a more compact representation of scene structure in cluttered RGBD images. Compared to shape models such as planes, cuboids etc., approximate convex shape model is more general and has more grouping power. With properly constructed energy function, the optimal grouping is able to give near-object level shapes or small number of object parts. When finding approximate convex shapes, we do not simply judge the result by the convexity of each component. For instance, a convex representation using single points is not desirable because it is not a minimum number representation. We are interested in the representation that satisfies overall low local concavity, small intersection, max-covering and min-number constraints. Convexity is one soft constraint in the formulation. Such a setting enables our method to group 3D object regions even with holes or other types of non-convexity.

The proposed method is related to approximate convex decomposition. Different methods have been proposed to decompose a 2D shape or a 3D mesh object to approximate convex components. Delaunay triangulation is a simple method of convex shape decomposition. However, decomposing a polyhedron into minimum number of convex components is NP-Hard [4]. A quadratic bound exists for an approximation algorithm [4]. Fortunately, exact convex decomposition is often not necessary and also not desirable in computer vision applications because very few objects are perfectly convex. A fast greedy merging method has been proposed in [5] for approximate convex decomposition. In [9], methods of finding good concavity cuts are proposed and as an extension in [10], a dynamic programming method is further used to optimize the cuts for the decomposition. Other global optimization methods have also been used in [8,6,7] to optimize these cuts with different extra constraints. These previous methods have been mostly targeted at graphics applications on clean 3D models. In computer vision, their application is mostly limited to 2D shape representation. Different from the previous shape decomposition approach, in this paper, we directly optimize the selection of a set of approximate convex shapes in RGBD images. Ideally, these shapes should represent whole objects that are roughly convex or the convex parts of concave objects. Finding convex objects is different from previous decomposition approaches: we allow small overlaps between shapes due to noisy observations. To our knowledge, there is no previous works that tackle convex object finding in RGBD images. We propose a fast and reliable solution.

Point cloud segmentation is also related to the proposed method. These methods either cluster the points using the affinity of point color and 3D coordinates or rely on semantic classifiers to categorize each point into an object class [13,14,15]. Object supporting relations are further estimated in [11]. A method that uses object bounding box spatial and supporting relations for object region segmentation in stereo images has been proposed in [12]. Different from these previous methods, the proposed method is more about shape finding than

segmentation and we allow small overlaps between shape detections. Our method also does not need to recognize the surface point classes in shape detection.

In summary, the contribution of the paper includes: we tackle the new problem of convex shape finding in RGBD images; we propose a new method to extract a set of roughly convex shape candidates from RGBD images; and, we propose a novel two-stage linear optimization method to find approximate convex shapes.

2 Methods

2.1 Overview

Our proposed method first extracts 3D shape candidates. There is no restriction that in these 3D shape candidates there is a disjoint set to partition the scene. We thus can use different candidate generation approaches to increase the success rate of shape detection. In this paper, to illustrate the concept, we use two low-complexity superpixel merging procedures to generate the candidates. Region proposals from other methods can be included to further improve the performance. Each shape candidate is represented as a 3D superpixel on the surface of a scene. Our goal is to select a subset of shape candidates that best describe the RGBD data and satisfy the geometrical constraints.

In more details, let $\mathbf{x}$ be the indicator vector of these shape candidates. Each binary element in the vector $\mathbf{x}$ indicates whether the corresponding shape candidate is selected. We try to optimize the following energy function.

$$\begin{aligned} &\min_{\mathbf{x}}\{U(\mathbf{x}) + \alpha N(\mathbf{x}) + \beta I(\mathbf{x}) - \lambda P(\mathbf{x})\} \\ &\text{s.t. } \mathbf{x} \text{ is a valid shape candidate selection.} \end{aligned} \tag{1}$$

Here $U(.)$ quantifies the costs of the selected candidates. This is a unary term. $U(.)$ measures the concavity of the shapes. By minimizing $U(.)$, the optimization tends to select roughly convex candidates. Term $N(.)$ penalizes the number of selected candidates; we try to select a small number of candidates to represent the scene. $I(.)$ is the intersection term that quantifies the intersection volume between the convex hulls of neighboring shapes and the intersection area of their projections on the image plane. Term $P(.)$ encourages the selected shapes to spread out and cover a large area of the 3D scene surface. Term P is critical; without it, the optimization gives a trivial all zero solution. P is a high order term that couples everything together. Constants α, β and λ control the weight among the terms. By optimizing the above energy function, we select an optimal set of shape candidates so that they are roughly convex, have little 2D and 3D intersection, have a small number and cover a large region of the 3D scene surface. In fact, our optimization contains two stages, each having slightly different objective function and constraints. The first stage is used to deal with the shapes whose convexity is within a range and the second stage is used to handle the rest of the shapes. The optimization contains high order term and is hard to solve directly. Naive exhaustive search is infeasible because we have a large set of candidates. In the following, we study how the formulation can be linearized and how we can solve the problem using an efficient branch and bound method.

2.2 Candidate Extraction

We first find a set of approximate convex shape candidates from the RGBD images. The candidates are essentially composed of a group of 3D points in the original point cloud. We use several different methods to generate these candidates to increase the chance that we include all the approximate convex shapes. Note that we do not require that these candidates all have high quality. The important factor is that the good ones are hopefully included in the candidate set if we generate enough candidate shapes. Through optimization, we make a more accurate final decision. We partition the 3D point clouds into over-segmented superpixels using both the geometry and color information. By merging these 3D superpixels iteratively using the convexity constraints, we generate a set of candidate approximate convex shapes.

The over-segmented superpixels are preferably small planar patches. We use two different approaches to generate these superpixels. The first one is a modification of the graph based segmentation method [16]. Instead of using color alone, we use both color and depth when computing the affinity of neighboring pixels. The x, y and z maps of a RGBD image are normalized to range [0,1]. The color image's three channels are also in the range [0,1]. The distance between two points is defined as the summation of the point and the color distance. With the combined distance, we obtain the superpixels using the graph based segmentation method [16]. Apart from the superpixels generated using the above method, we also use a k-means method to generate over-segmented patches. Each image pixel corresponds to a vector $(n_x, n_y, n_z, d, \bar{z})$, where (n_x, n_y, n_z) is the normalized normal vector of the 3D scene surface at point (x, y, z). Here $d = xn_x + yn_y + zn_z$, the signed distance of the plane passing (x, y, z) with normal (n_x, n_y, n_z) to the origin. $\bar{z}$ is the normalized z in range [0,1]. The k-means clustering on the 5D vectors generates K clusters, e.g. $K = 20$. Note that the number of clusters is different from the number of generated superpixels. Since a cluster from k-means may contain multiple disconnected components, the effective superpixels after we extract connected components are many more than the number of clusters. An example in Fig.3 (c,d) shows the superpixels generated using these two approaches. As shown in the figure, over-segmented superpixels are able to capture the object boundaries quite well. The k-means method essentially clusters image pixels into flat 3D patches. The graph segmentation method uses both the color and the geometric information in the clustering. These over-segmented image patches are usually not able to yield masks for heterogeneous large targets. We therefore successively merge the superpixels to generate more shape candidates. If properly implemented, the merging procedure has a high chance to capture these bigger convex structures.

The merging procedure is built on top of the concavity measurement. Different from a complete mesh model, our mesh model is an incomplete surface that has a "front" side and a "back" side. A 3D superpixel is roughly convex if the shape extrudes towards the front, otherwise it is roughly concave. We define the front and back side of the surface using a ray starting from the camera center: (0,0,0) in the camera coordinate system. The segment of the ray from (0,0,0) to a point

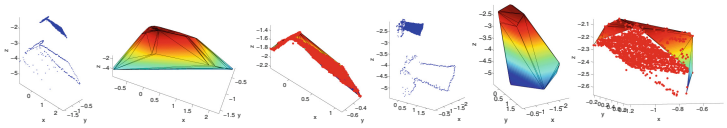


Fig. 2. (a): 3D point set with pushed-away boundaries. Here boundary points are scaled by two. (b): The convex hull. (c): The 3D points on the shape with the frontal hull. (d)-(f) show another concave shape example of measuring the concavity.

in the 3D point cloud is the front and the rest part of the ray is the back of the surface. Note that the seemingly simple way of defining the front and back using an orthographic projection is wrong; there is no guarantee that a ray parallel to the optical axis would have only a single intersection point with the scene surface.

The convexity measurement is obtained as follows. We first extend the boundary points of a 3D superpixel further away from the camera projection center (0,0,0). This can be easily implemented by multiplying a large scaling factor, e.g. 10, to the x, y and z of these boundary points. We then compute a 3D convex hull of this modified 3D superpixel. The next step is to remove the 3D triangles on the convex hull with at least one vertex on the modified boundary points. By removing these patches, we are left with an frontal hull of the 3D superpixel. Fig. 2 shows the procedure. As shown in Fig. 2, if a 3D superpixel is roughly convex, the mean distance of the points to the frontal hull is small; otherwise as shown in Fig. 2 (f), the mean distance of the points to the frontal hull is large. The mean distance of points to the frontal hull is thus used as the measurement of the convexity/concavity of a 3D superpixel. Apart from the 3D concavity, we also measure the concavity of the projected 2D superpixel on the image plane. The 2D concavity is defined as $1-N_f/N_c$, where N_f is the number of foreground pixels on the 2D superpixel and N_c is the number of pixels on the convex hull of the 2D superpixel. Both of the two measurements are used in the final shape selection.

With the 3D concavity measurement, we successively merge the neighboring superpixels. Two 3D superpixels are deemed as neighbors if their minimum 3D point distance is less than some threshold. In the merging procedure, two neighboring 3D superpixels that have the smallest concavity if combined is merged into a single superpixel. This procedure continues until there is only one superpixel left or all the superpixels are isolated. The shape candidates are composed of the 3D superpixels from the two partition methods and the new superpixels generated in the merging procedure. Since there is no sub-optimal condition for either of the merging process, we cannot just set a stopping criterion and get the optimal set of convex shapes using the simple merging procedure. Even though there is no optimal guarantee, there is still a high chance that the good convex shapes that should appear in the final shape selection exist in the merging procedure. In fact, there is no guarantee that the concavity score increases as the

patches become bigger. We thus select these shapes to form the approximate convex representation to minimize the energy function defined in Eq. (1).

2.3 Formulation

The formulation in Eq. (1) is a hard combinatorial problem due to the high order term that couples the candidate selection. Here we propose a two-stage linear formulation that can be efficiently solved using a branch and bound method. In the following, we show how each term can be linearized. Recall that we use a vector of binary variables $(x_1, x_2, ..., x_n)$ to indicate the selection of each candidate shape, where n is the number of 3D shape candidates.

Unary Term. Let $u_i = ag_iN_i + bh_i + cg_i$, where a, b and c are constant coefficients, g_i and h_i are the 3D concavity and 2D concavity and N_i is the number of points in the shape candidate i. Recall that g_i is the average distance of the points on the 3D superpixel to the frontal hull; g_iN_i is the total distance of the points to the frontal hull. We define the overall convexity term as $\sum_{i=1..n} u_ix_i$. When we minimize the unary term, we minimize the total distance of the points on the selected candidates to the local frontal hull, and the selected shapes also should have small concavity individually in both 2D and 3D.

We deal with the shape candidates with concavity higher than some threshold differently from those with lower concavity. Essentially, we want to assign the shape candidates whose concavity is within the range first and then determine the selection of the rest of the candidates. Due to camera distortion, large flat shape candidates such as wall and floor often appear curved (concave shaped) when they are far away from camera. These shape candidates may have large concavity measurement in RGBD images. When this happens, a single stage optimization has a hard time to balance the local shape cost and the global constraint costs no matter how we set the parameters in the energy function. We thus use a two-stage optimization to alleviate this problem. At the first stage optimization u_i is defined above, and at the second stage we only use g_iN_i in the unary term.

Binary Term. A physically feasible selection of 3D shape candidates should have small intersection to each other. Here we use the 3D convex hull intersections of these candidates to quantify how the 3D shapes interact to each other at the first stage of optimization. The intersection of two convex hulls is another convex hull. We represent the convex hull as a polytope: a convex hull can be represented as the intersection of a set of half spaces (the space on one side of a plane). Computing the exact intersection of two polytopes has high computational complexity. Since we only need a ratio to measure the intersection, we obtain the intersection ratio using the following space quantization method. We discretized the 3D finite visible volume in the camera frame of the RGBD image. Let $\{A_iw \leq b_i, i \in I_p\}$ represent the polytope of shape candidate p and $\{A_iw \leq b_i, i \in I_q\}$ the polytope of candidate q. Then the 3D

convex hull intersection ratio $r_{p,q} = \sum_{w \in V} s((\cap_{i \in I_p}(A_i w \leq b_i)) \cap (\cap_{i \in I_q}(A_i w \leq b_i)))/\sum_{w \in V} s((\cap_{i \in I_p}(A_i w \leq b_i)) \cup (\cap_{i \in I_q}(A_i w \leq b_i)))$, where V contains all the discretized voxel centers; $s(X) = 1$ if X is true and otherwise 0. The intersection constraint can be formulated as the following binary energy term $\sum_{\{p,q\} \in \mathcal{N}} r_{p,q} x_p x_q$, where $\mathcal{N}$ is the set of all neighboring 3D superpixels. We define two 3D superpixels are neighbors if their minimum point distance is less than a threshold. The intersection term is quadratic. We use a linear programming trick to linearize it. Since the assignment variable x is binary, we let $y_{p,q} = x_p x_q$ with the constraint that $y_{p,q} \geq 0$, $y_{p,q} \geq x_p + x_q - 1$, $y_{p,q} \leq x_p$, $y_{p,q} \leq x_q$. It is easy to verify that $y_{p,q}$ is 1 if and only if x_p and x_q are both 1, and otherwise 0. Therefore, by introducing the auxiliary variable $y_{p,q}$ the intersection term can be converted to linear term $\sum_{\{p,q\} \in \mathcal{N}} r_{p,q} y_{p,q}$ with the constraints on y. We also require that the intersection does not exceed some hard threshold R, i.e., $y_{p,q} = 0$ if $r_{p,q} > R$.

The 3D convex hull intersection constraint is only used at the first stage of optimization. At the second stage, we use the intersection ratio of the projected 2D superpixels on the image plane, $r_{p,q} = l_{p,q} = \mathcal{A}(S_p \cap S_q)/\mathcal{A}(S_p \cup S_q)$, where $\mathcal{A}(.)$ is the area of a region and S_p and S_q are the projected 2D superpixel regions of the 3D shapes on the image plane.

Global Term. It is not enough to use the above unary and binary terms alone. Since all the coefficients in the unary and binary terms are positive, zero vector is a trivial solution. We need a global term to enforce that enough shapes should be selected. We cannot constrain the number of the selection because in most of the cases this number is unknown. A natural choice is to enforce that the set of chosen shape candidates should cover most of the image plane if projected. This in fact is equivalent to let the shapes cover as much of the 3D points as possible since each 3D point corresponds to a single 2D pixel in the image. Note that the seemingly simpler constraint that each point of the image should belong to a selected shape is not always feasible. The max-covering constraint is necessary.

The problem size would be too big if we model the covering term at the point level. Instead, we quantize the image plane into small rectangle patches. For 480×640 images, 20×20 tile is sufficient. A shape candidate covers a tile if the projection of the shape candidate on the image plane has an intersection with the tile. We introduce auxiliary variable z_k to indicate whether tile k is covered: $z_k = 1$ if tile k is covered by at least one shape candidate. The tile covering variable z is related to the candidate selection variable x by: $\sum_{i \in \mathcal{F}_k} x_i \geq z_k$, $0 \leq z_k \leq 1$, where $\mathcal{F}_k$ is the set of shape candidates that cover tile k. To encourage max-covering, we just need to maximize the overall covering $\sum_{k \in \mathcal{T}} z_k$, where $\mathcal{T}$ contains indexes of all the small tiles. In fact, we do not have to constrain z to be binary. We can verify that if at least one selected shape candidate covers tile k, z_k has to be 1 to maximize the covering term; if all the shape candidates that cover tile k are not selected, z_k has to be 0. $\sum_{k \in \mathcal{T}} z_k$ indeed quantifies the amount of covering.

Number Term. By enforcing the max-covering term, we avoid the trivial zero solution problem. However, there is still a bias for the algorithm to select many small shape candidates. Notice that since we multiply the number of points to concavity, we in fact tend to select many small shape candidates to represent the whole scene and this may result in a smaller overall point distance to the local frontal hulls. The other two terms in the convexity term can relieve the problem a little bit. Along with these two terms, we explicitly introduce another number term $\sum_i x_i$ to the objective function to penalize the selection of a large number of shape candidates. By adjusting the weight of this term, we are able to select a relatively small number of large shape candidates to explain the whole scene.

The complete two-stage linear formulation is as follows. At the first stage, we optimize:

$$\min \sum_i (ag_i N_i + bh_i + cg_i + \alpha)x_i + \beta \sum_{\{p,q\}\in\mathcal{N}} r_{p,q} y_{p,q} - \lambda \sum_{k\in\mathcal{T}} z_k$$

$$\text{s.t. } y_{p,q} \geq 0,\ y_{p,q} \geq x_p + x_q - 1,\ y_{p,q} \leq x_p,\ y_{p,q} \leq x_q,\ \forall \{p,q\} \in \mathcal{N}$$

$$y_{p,q} = 0, \text{ if } r_{p,q} > R$$

$$\sum_{i\in\mathcal{F}_k} x_i \geq z_k,\ 0 \leq z_k \leq 1,\ \forall k \in \mathcal{T}$$

$$x_i = 0, \text{ if } g_i > G \text{ or } h_i > H,\ x_i = 0 \text{ or } 1$$

where G and H are thresholds for the 3D and 2D concavity. The first stage of optimization determines the dominant approximate convex shapes. If we just use this optimization, some large shapes that are not convex enough due to the camera distortion will be left unlabeled. The second stage optimization relaxes the labeling criterion. It assumes the labels on the shape candidates that are labeled 1 at stage one fixed but only tries to label the rest of the candidates. The second stage uses a more relaxed 2D intersection measurement and removes the range constraints on x.

$$\min \sum_i (a' g_i N_i + \alpha')x_i + \beta' \sum_{\{p,q\}\in\mathcal{N}} l_{p,q} y_{p,q} - \lambda' \sum_{k\in\mathcal{T}} z_k$$

$$\text{s.t. constraints on } y \text{ and } z \text{ are the same as stage one}$$

$$x_i = 1 \text{ if they are labeled as 1 at the first stage, } x_i \text{ is binary variable}$$

Parameters Training. There are quite a few parameters that need to be determined in the objective function and constraints. The hard constraint parameters G and H can be directly obtained from the training data. They are set such that they are the upper bounds of g_i and h_i for all the positive training data. The intersection bound R is set to 0.5. To obtain the parameters $a, b, c, \alpha, \beta, \lambda$ in the objective function, we optimize a linear program. The idea is that we need to select a set of parameters such that the objective value is less than that of the sub-optimal shape candidate selection. We solve the following linear program to obtain a good set of parameters.

$$\min_{a,b,c,\alpha,\beta,\lambda} M\sum_{j=1}^{J}(a\xi_a^j + b\xi_b^j + c\xi_c^j + \alpha\xi_\alpha^j + \beta\xi_\beta^j - \lambda\xi_\lambda^j) -$$

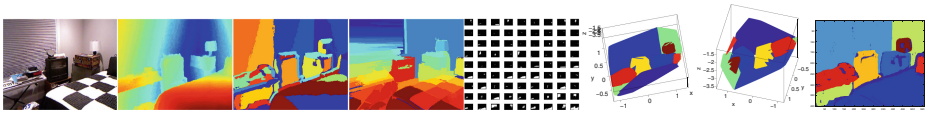
$$\sum_{j=1}^{J}\sum_{m=1}^{M}(a\phi_a^{j,m} + b\phi_b^{j,m} + c\phi_c^{j,m} + \alpha\phi_\alpha^{j,m} + \beta\phi_\beta^{j,m} - \lambda\phi_\lambda^{j,m})$$

$$\text{s.t. } a, b, c, \alpha, \beta, \lambda, \geq 0,\ a + b + c + \alpha + \beta + \lambda = 1.$$

Here ξ are the coefficients of $a, b, c, \alpha, \beta, \gamma$ in objective function for ground truth data; we have J RGBD images with ground truth approximate convex shape selections. ϕ are the corresponding coefficients in the objective function using randomly generated shape candidate selections in RGBD images; random labeling is repeated M times for each RGBD image. By optimizing the above energy function, we select a set of parameters that give low objectives on the ground truth labeling and high values on the negative samples. Note that we need to optimize the coefficients for both the first stage and second stage linear formulation. We sequentially find these coefficients for stage one and then stage two. The linear program can be solved efficiently using the simplex method or the interior-point Method.

2.4 Optimization

By converting the shape candidate selection into a linear formulation, we can efficiently solve the optimization using a branch and bound procedure. Our formulation is a mixed integer linear program, in which x is binary and the rest of the variables are floating point variables. We first solve the linear program by discarding the integral constraints. If all x are integral, we obtain the global optimal solution. Otherwise, we select x_i that is closest to 0.5 and generate two branches: left branch with $x_i = 0$ and right branch with $x_i = 1$. From the floating point solution of the linear program, we obtain the first guess of the all integer solution for x by rounding x to the closest integer. Other variables' value can be obtained from x. If $x_i = 0.5$ a random selection of 0 or 1 is used to break the tie. Using this first guess, we obtain the first upper bound of the objective function. For each branch, we re-solve the linear program and obtain a lower bound for each branch. If the lower branch is greater than the current upper bound, the branch is pruned. If there is still floating point x, we keep the current branch active. We use the rounding method to re-estimate the upper bound. If the estimated upper bound is lower than the current one, we update the upper bound. Among all the active branches, we choose the one with the lowest lower bound and branch on the most ambiguous x variable. This procedure continues until the lowest lower bound of active branches equals the current upper bound or some pre-defined tolerance gap is achieved. In this paper, the ratio of the tolerance gap to the upper bound is set to be 10^{-6}. For a typical problem with the number of shape candidates around 500. The branch and bound procedure is surprisingly fast: it takes one or two seconds to converge to the global optimal solution for each stage of the candidate selection.



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

Fig. 3. An example. (a): Color image. (b): The aligned depth map. (c): Superpixels from k-means. (d): Superpixels from graph method [16] using both the color and depth image. (e): Sample candidate masks. (f, g): The convex shapes from the proposed method in two different view points. (h): The masks of the shapes on the image plane.

3 Experiments

Fig. 3 shows an example about how the proposed method works in finding approximate convex shapes in a RGBD image. The test image is from NYU RGBD dataset [11] and we captured our 100 sample images using Kinect sensor for parameter training. The parameters are fixed in all the following tests. The proposed method first converts the RGBD image into a color point cloud. As shown in Fig. 3, two over-segmented superpixel maps are extracted using both the color and depth maps. The two maps contain 82 and 162 candidates respectively. The superpixels are then merged successively based on their convexity from the view point to generate a total of 486 candidate shape masks. The optimal set of shape candidates is then chosen using the proposed two-stage linear method. The first stage optimization has 44192 rows (constraints) and 15209 columns (variables). The relaxed linear programs takes 973 steps to converge. The search tree branch and bound only takes one more step since the linear program gives a tight lower bound. The first stage optimization takes 1.3 seconds in a 2.8GHz machine. The second stage optimization has 59463 rows and 20565 columns; it takes just one step to find the global optimum. The convex hulls of these shapes and the shape masks on the image plane are shown in Fig. 3. Our proposed method reliably finds the approximate convex shapes in RGBD images. It is also efficient. The whole procedure takes few seconds.

We further compare the proposed method with HACD (Hierarchical Approximate Convex Decompose) [5], a greedy approach that successively merges regions until some concavity tolerance is reached. HACD is a typical implementation of approximate convex decomposition. HACD is open-source and has been shown to work very well on different graphical mesh models. Its concavity measurement makes it also suitable for finding convex objects on a surface that is a single view of the scene. We adjust the parameters for HACD to achieve the best results. Fig. 4 shows the comparison results of the proposed method against HACD. HACD makes quite a few mistakes in convex decomposition of RGBD meshes: many large objects such as a table and a floor have been split into parts and some convex parts are not separated out. Our proposed method greatly improves the results. In terms of computational complexity, our method is also hundreds of times faster than HACD.

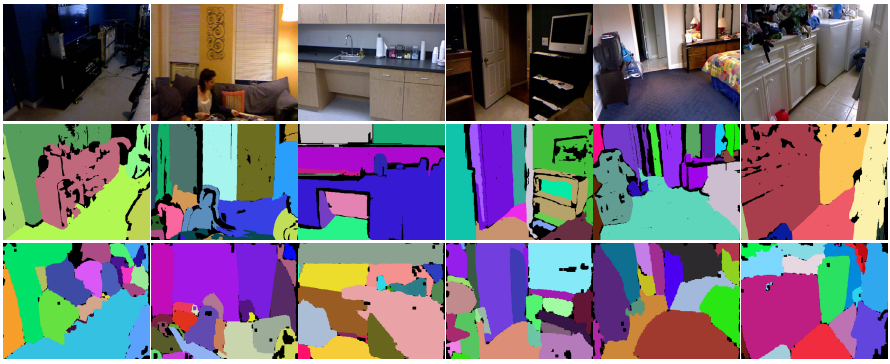
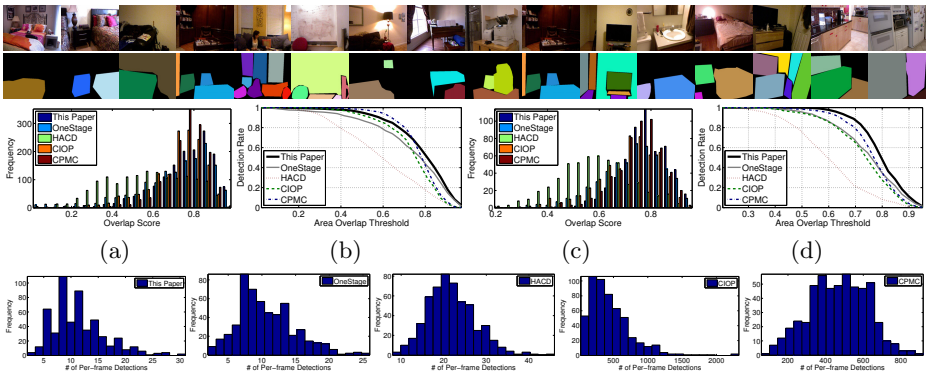


Fig. 4. Comparison between the proposed method and HACD [5]. Row 1: Color images. Row 2: Near-convex 3D shapes found by the proposed method. We show the projections of these shapes to the image plane. Row 3: Near-convex shapes found by HACD [5].

Beyond qualitative results, we further quantify the results by comparing against the ground truth. The original object region ground truth labeling in the NYU dataset [11] is not quite suitable because it contains many non-convex shapes and the labeling often decomposes objects into plane surfaces instead of the convex parts we are most interested in. We labeled 538 images in the NYU dataset and 1471 approximate convex shapes in these images. Some samples of the ground truth shape masks on the images are shown in Fig. 5. With the ground truth labeling, we quantify the performance of different convex shape finding methods using matching score distributions and detection curves. For each ground truth convex shape mask, we go through all the convex shapes found by different methods and if the overlap between the ground truth mask and the shape mask detected is above a threshold we have a successful detection. The overlap between two masks is defined as the ratio of the intersection mask area to the union mask area. Fig. 5 shows the overlap score distributions and the detection curves of the proposed method and HACD. Two criteria are used: one is the per-object overlap score distribution and detection rate, and the other one is the per-frame overlap score distribution and detection rate. The per-frame mask matching score is defined as the average overlap score for all the ground truth objects in an image. If the per-frame overlap score is above a threshold, we deem the frame correctly processed. For better performance, overlap score distribution should have large portion of the curve focused on the right and small tail on the left; the better detection curve gives higher detection rate for each fixed region overlap threshold. Our proposed method gives significantly better results in both the per-object and per-frame test cases than HACD. We also test using just the first stage of the optimization in our method. As shown in Fig. 5, the proposed two-stage approach gives much better result than the one-stage method.

Our method belongs to the class of methods that are based on region proposals. We further compare the proposed method with other methods that are based



	This Paper	OneStage	HACD [5]	CIOP [20]	CPMC [19]
Per-frame mean detection rate	0.7816	0.7282	0.5835	0.7104	0.7477
Per-obj mean detection rate	0.7641	0.7109	0.5729	0.7150	0.7537
Mean # of per-frame detections	11.35	10.72	21.84	448.6	467.8

Fig. 5. Row one and two show sample ground truth labeling of approximate convex objects in the RGBD images from the NYU RGBD dataset [11]. OneStage is the method that uses only the first stage optimization of the proposed method. (a): Per-object shape matching score distribution. (b): Per-object detection rate. (c): Mean shape matching score per-frame distribution. (d): Per-frame detection rate. (e)-(i): The per-frame detection number histograms of different methods.

on the region candidates using category independent object proposals (CIOP) [20] or CPMC [19]. These methods' detection rate limit is set by the region candidate proposals. We compare the proposed method with the limit detection rates of competing methods using shape candidates generated from [20] or [19]: if proposal candidate matches the ground truth we deem it is a successful detection. For fair comparison, we modified the code of CIOP and CPMC to use both color image and the depth image for more reliable proposal generation in RGBD images. The per-object and per-frame comparisons of the overlap score distributions and the detection rates are shown in Fig. 5. The per-object detection rate limit of CPMC based methods is a bit higher than the proposed method if the threshold is lower than 0.75. However, the detection rate with threshold greater than 0.75 is what we really care about. Our method gives better results in all the other test cases. The average detection rate of the proposed method is also the highest as shown in Fig. 5.

We have applied the proposed method to find approximate convex objects in all the 1449 RGBD images in the NYU dataset. Fig. 6 shows sample results. Our method reliably detects approximate convex targets in cluttered RGBD images. Failure cases are due to holes and noisy depth data on some object surfaces. The unreliable and noisy data may confuse the convexity estimation and causes missing the true candidate proposal. By further improving the region proposal generation against noise or using sensors with low noise we are able to further improve the results.

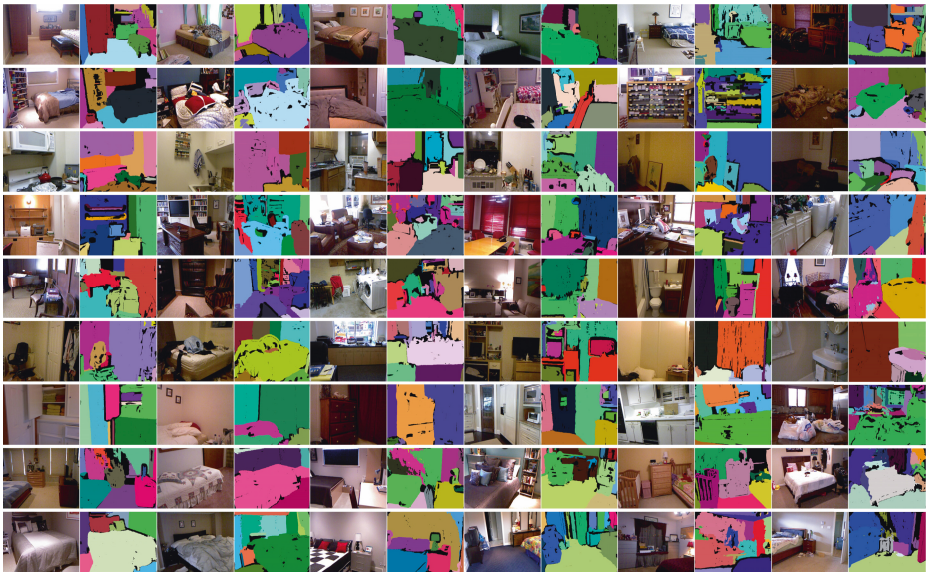


Fig. 6. Random sample results of the proposed methods on the NYU RGBD dataset [11]. Odd columns show the color images and even columns show the approximate convex object regions projected to the image plane. Our proposed method reliably detects approximate convex objects in cluttered RGBD images.

4 Conclusion

Finding approximate convex shapes in RGBD images is a new task. In this paper, we propose a novel global method to tackle the problem. We formulate the optimization into a two-stage mixed integer program that selects the shapes from a large set of candidates. An efficient branch and bound method is applied to solve the two-stage optimization. Our evaluation results on thousands of RGBD images show that the proposed method is reliable, fast and more accurate than the competing methods.

Acknowledgment. This research is supported by the U.S. NSF funding 1018641.

References

1. Schnabel, R., Wessel, R., Wahl, R., Klein, R.: Shape Recognition in 3D Point-Clouds. In: WSCG 2008 (2008)
2. Li, Y., Wu, X., Chrysathou, T., Sharf, A., Cohen-Or, D., Mitra, N.J.: GlobFit: Consistently Fitting Primitives by Discovering Global Relations. In: ACM SIGGRAPH (2011)
3. Leonardis, A., Jaklic, A., Solina, F.: Superquadrics for Segmenting and Modeling Range Data. TPAMI 19(11) (1997)

4. Chazelle, B.: Convex Partitions of Polyhedra: A Lower Bound and Worst-Case Optimal Algorithm. SIAM J. Comput. 13, 488–507 (1984)
5. Mamou, K., Ghorbel, F.: A Simple and Efficient Approach for 3D Mesh Approximate Convex Decomposition. In: ICIP (2009)
6. Ma, C., Dong, Z., Jiang, T., Wang, Y., Gao, W.: A Method of Perceptual-based Shape Decomposition. In: ICCV (2013)
7. Ren, Z., Yuan, J., Li, C., Liu, W.: Minimum Near-Convex Decomposition for Robust Shape Representation. In: ICCV (2011)
8. Liu, H., Liu, W., Latecki, L.J.: Convex Shape Decomposition. In: CVPR (2010)
9. Lien, J.M., Amato, N.M.: Approximate Convex Decomposition of Polyhedra. In: ACM Symposium on Solid and Physical Modeling (2007)
10. Ghosh, M., Amato, N.M., Lu, Y., Lien, J.M.: Fast Approximate Convex Decomposition Using Relative Concavity. Computer-Aided Design archive 45(2) (2013)
11. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor Segmentation and Support Inference from RGBD Images. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part V. LNCS, vol. 7576, pp. 746–760. Springer, Heidelberg (2012)
12. Bleyer, M., Rhemann, C., Rother, C.: Extracting 3D Scene-Consistent Object Proposals and Depth from Stereo Images. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part V. LNCS, vol. 7576, pp. 467–481. Springer, Heidelberg (2012)
13. Koppula, H.S., Anand, A., Joachims, T., Saxena, A.: Semantic Labeling of 3D Point Clouds for Indoor Scenes. In: NIPS 2011 (2011)
14. Silberman, N., Fergus, R.: Indoor Scene Segmentation Using a Structured Light Sensor. In: ICCV Workshop on 3D Representation and Recognition (2011)
15. Gupta, S., Arbelaez, P., Malik, J.: Perceptual Organization and Recognition of Indoor Scenes from RGB-D Images. In: CVPR 2013 (2013)
16. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient Graph-Based Image Segmentation. IJCV 59(2) (2004)
17. Jiang, H., Xiao, J.: A Linear Approach to Matching Cuboids in RGBD Images. In: CVPR 2013 (2013)
18. Lin, D., Fidler, S., Urtasun, R.: Holistic Scene Understanding for 3D Object Detection with RGBD Cameras. In: ICCV 2013 (2013)
19. Carreira, J., Sminchisescu, C.: Constrained Parametric Min-Cuts for Automatic Object Segmentation. In: CVPR 2010 (2010)
20. Endres, I., Hoiem, D.: Category Independent Object Proposals. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 575–588. Springer, Heidelberg (2010)