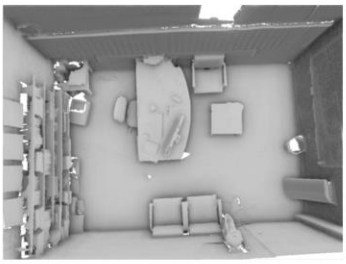
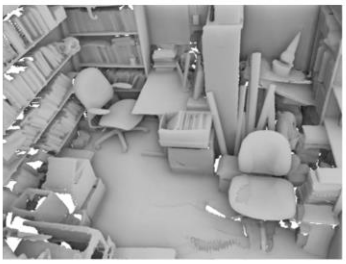
Designing a Robust Interactive Tool for 3D Scene Annotation

Creating and Understanding 3D Annotated Scene Meshes

Hi everyone, thank you for coming this presentation!

Motivation

- High-quality 3D scenes using RGB-D cameras are widely available.

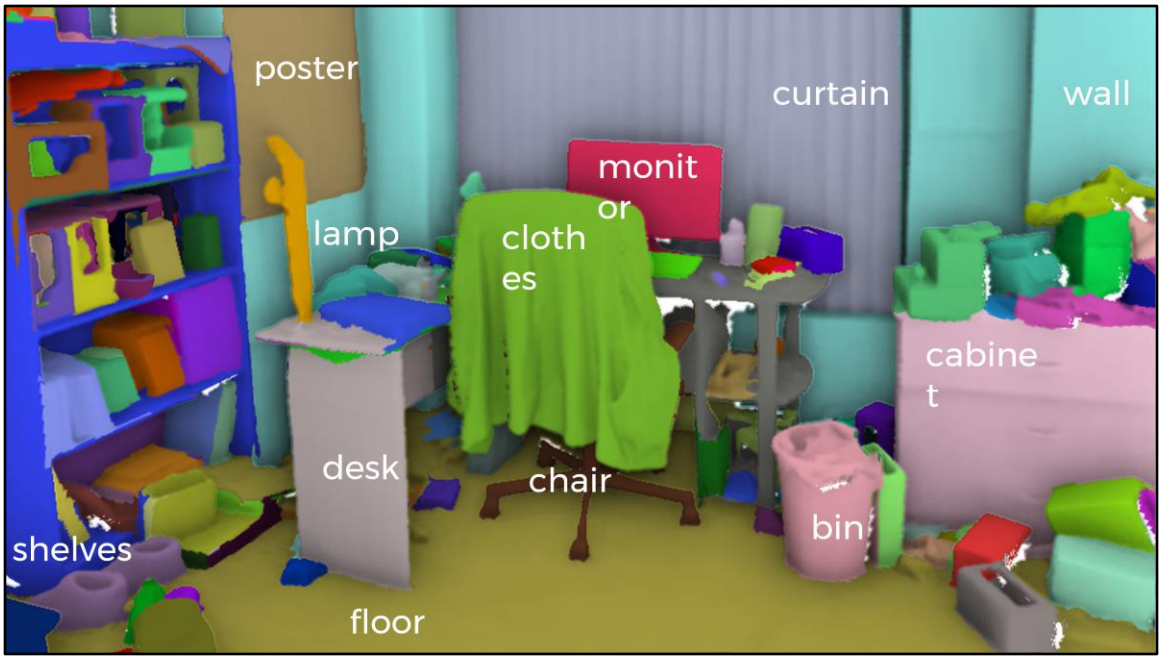




2

From the perspective of a computer or a robot, 3D reconstructed data is just a bunch of 3D vectors in XYZ coordinates.

This is very different from human perspective. For example, we understand and reason about the 3D content. We know what and where a particular object is.

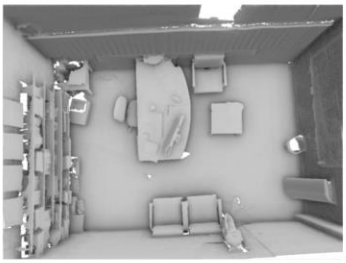
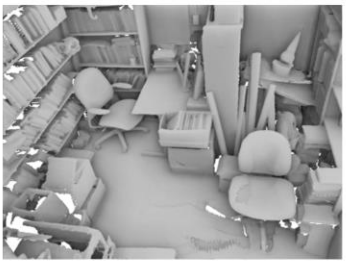
If you recall the topic of this talk, they are acquisition, understanding, and remodelling.



Scene understanding is the goal we pursue after 3D reconstruction.

Motivation

- Semantic segmentation, object detection, pose estimation are still challenging to solve for 3D scenes.

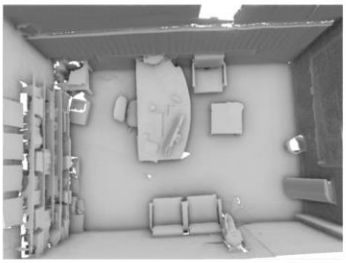
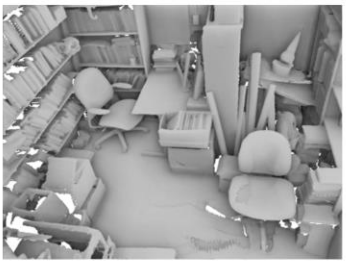




5

Motivation

- Deep learning needs massive ground truth data for training.

How to annotate 3D scenes effectively?

6

Problem statement

An interactive tool for scene segmentation and annotation

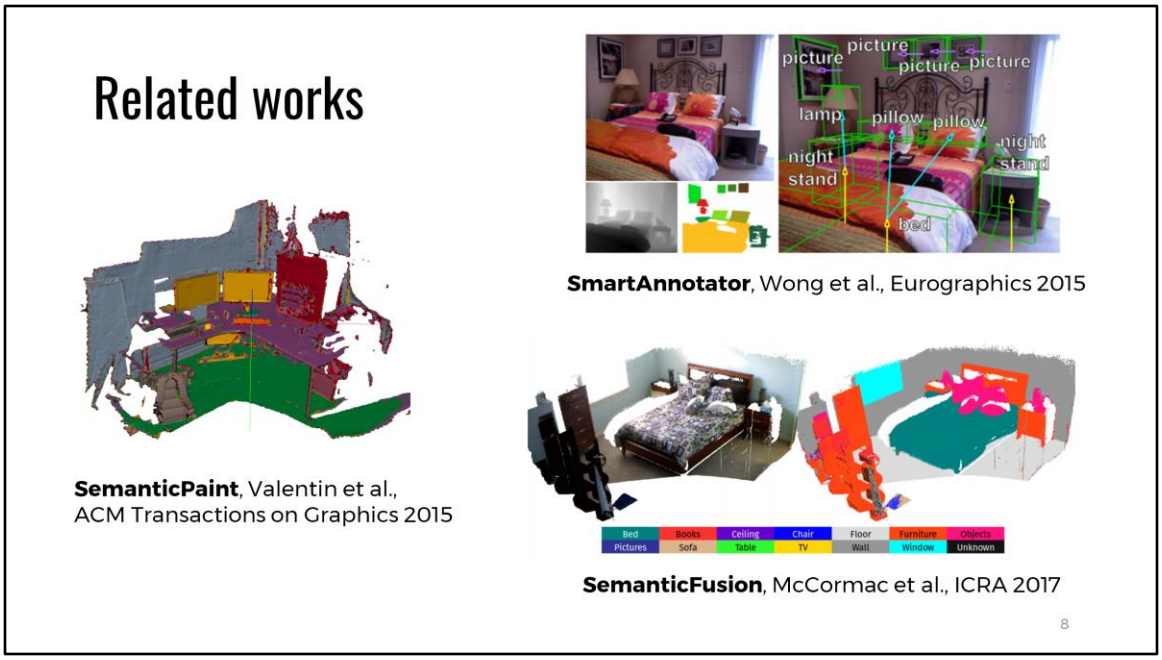
- Annotate a 3D scene and many RGB-D images in a single system.
- No world assumption. Capable to annotate any scenes.
- Dense annotation: per vertex and per pixel label.
- Fine-grained annotation: object poses, bounding boxes.

7

Interaction is required because no automatic segmentation is perfect.

We would like to be able to annotate on both 3D and 2D domain.

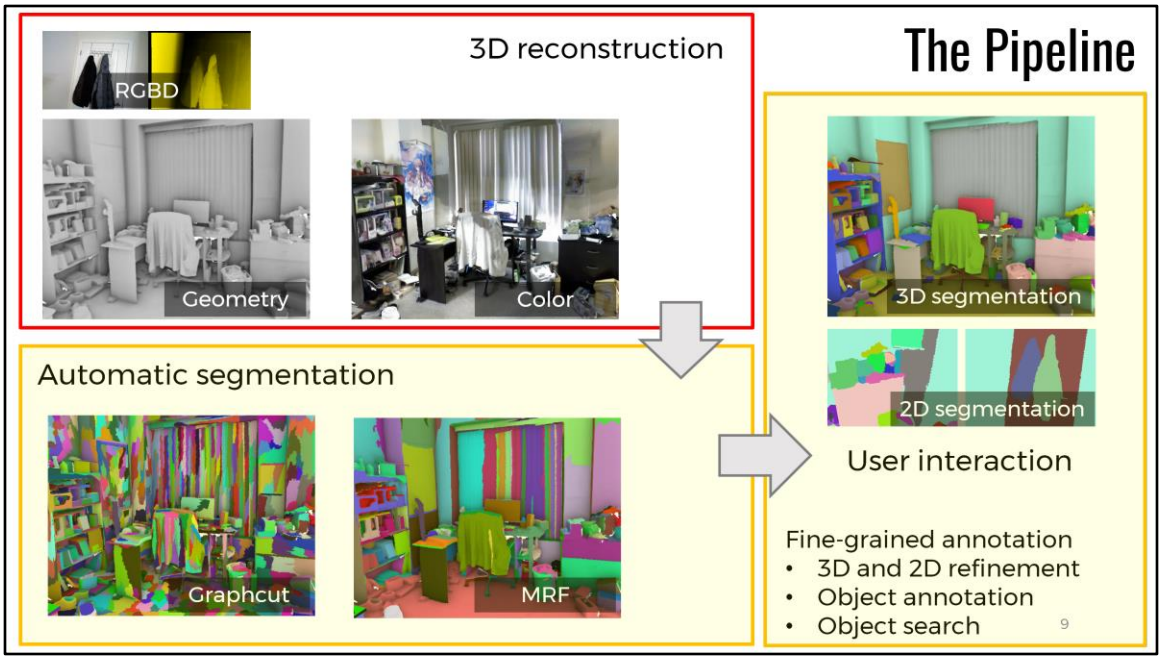
We make no assumption about the scene to annotate, e.g., Manhattan world. Our goal is to support arbitrary scenes from multi-sources, e.g., scenes from RGB-D reconstruction or multi-view geometry.



There are a few related works in this area.
SemanticPaint by Valentin et al. applied online learning and conditional random field to propagate labels from user interaction automatically.
Their system couples 3D reconstruction and segmentation together in real-time.

SemanticFusion published in ICRA recently also addressed scene segmentation and annotation, but their method is fully automatic, which is more useful for robot navigation and exploration than making data for deep learning.

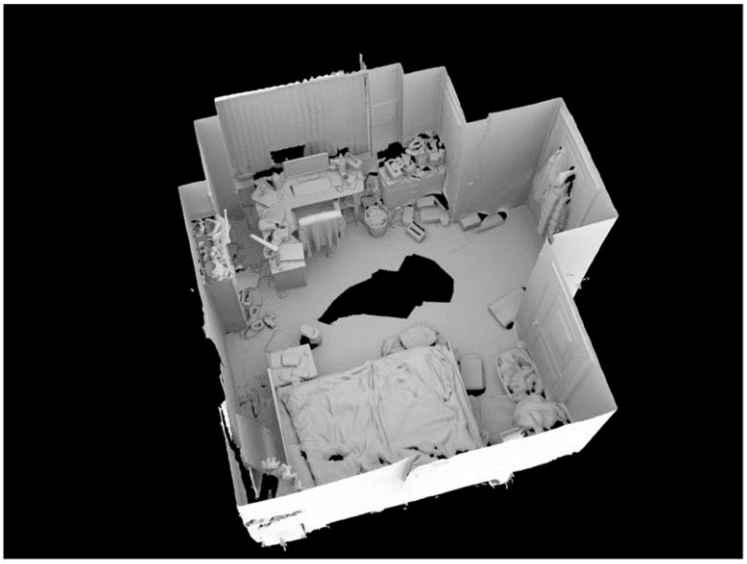
SmartAnnotator by Wong et al. in Eurographics 2015 performs segmentation and box annotation, and their system only worked with images.



Our proposed system has three main components: 3D reconstruction, automatic segmentation, user interaction.

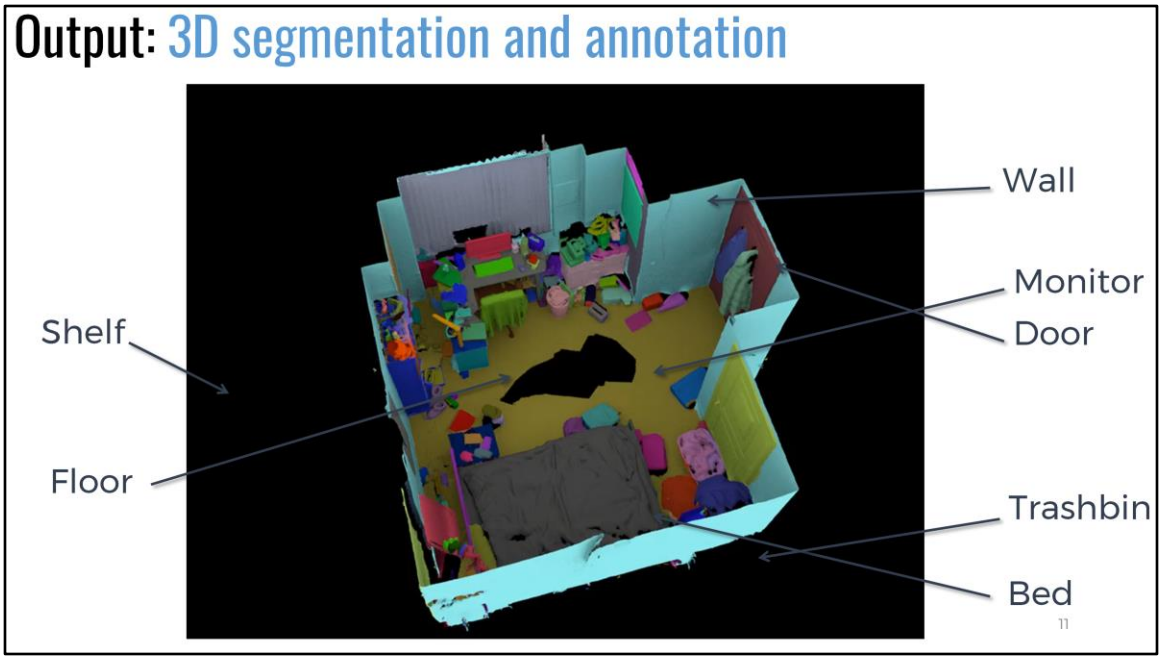
So here we will focus on automatic segmentation, and user interaction. We will also present advanced features such as 2D refinement and object search to support user to annotate more efficiently.

3D reconstruction



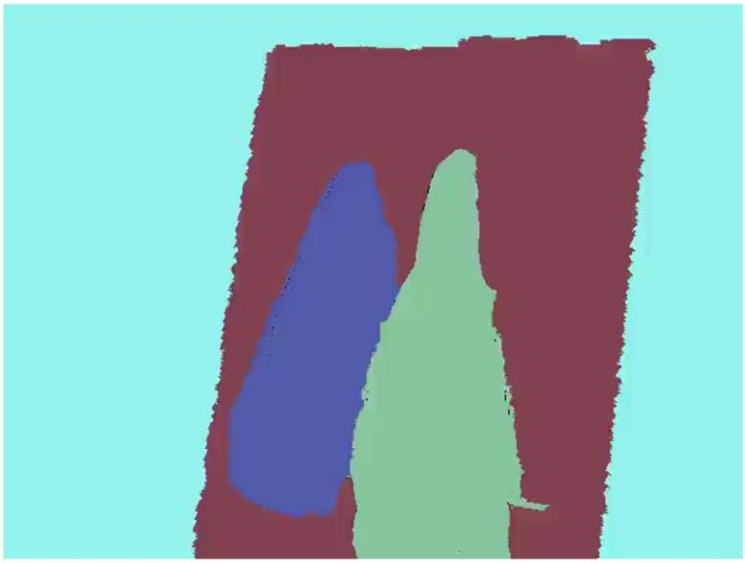
10

We then reconstruct the scene from the input video. We represent the scene as a triangular mesh.



Our system outputs a 3D mesh with the segmentation and annotation results.

Output: 2D segmentation and annotation

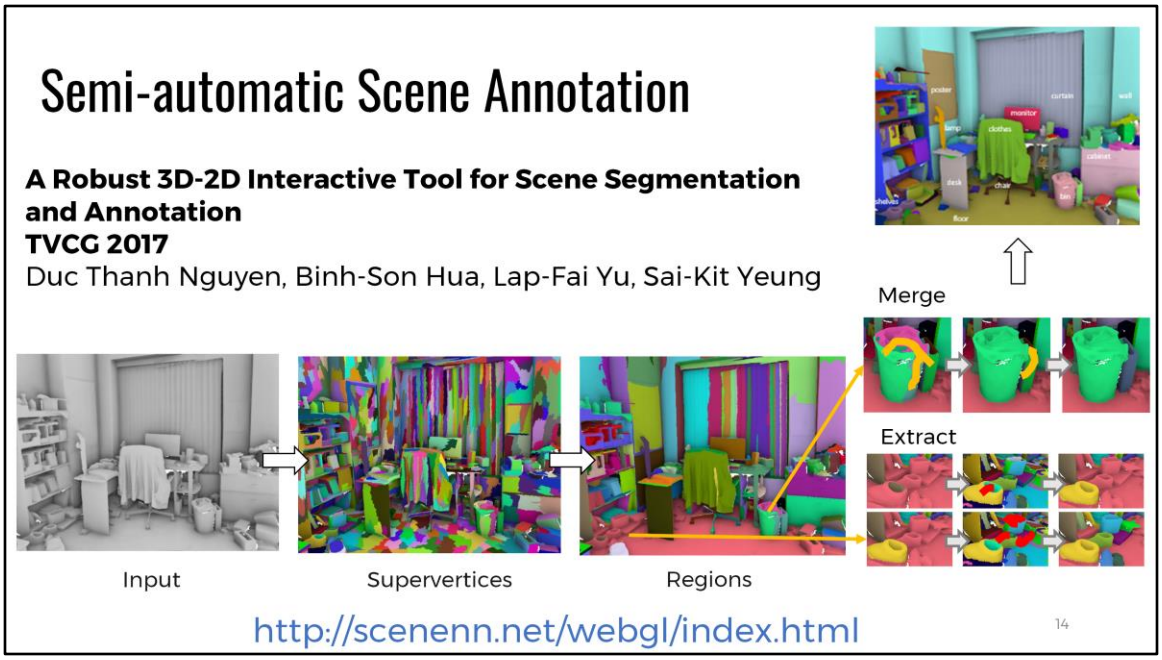


12

Similarly, our system also outputs a video by reprojecting the annotation from 3D to 2D.

3D segmentation

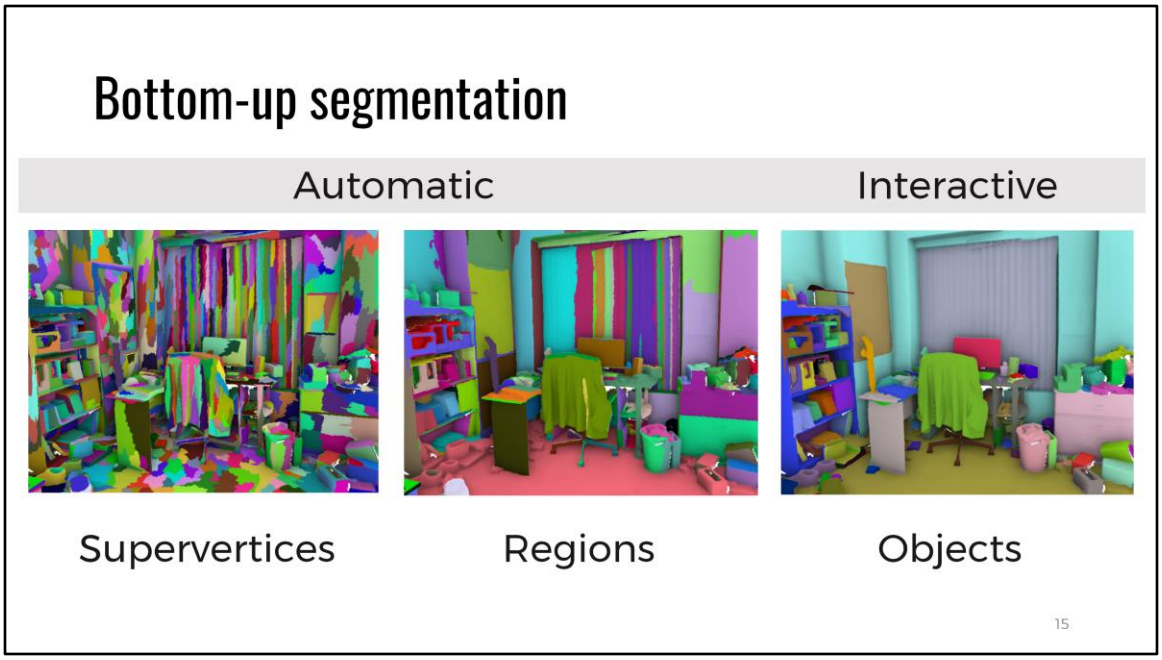
13



To support scene understanding and evaluate its performance, we need a set of ground truth data, which we obtain via a semi-automatic scene annotation tool.

Our input mesh is obtained from marching cubes after KinectFusion and global registration. Typical number of input vertices for each scene ranges between 1-2 millions. Direct manipulation millions of vertices is very challenging for a user interactive application. User usually cannot manage such a great deal of vertices.

For object segmentation, we assume that we only need to manipulate segment size at least close to some common small objects in indoor scenes such as cup, mouse, mobile phones, shoes, etc. With this assumption, we design a bottom-up segmentation system. We first apply automatic segmentation algorithms to generate supervertices and regions. Users can then manipulate supervertices and regions with a set of simple operations to segment objects.



Our input mesh is obtained from marching cubes after KinectFusion and global registration. Typical number of input vertices for each scene ranges between 1-2 millions. Direct manipulation millions of vertices is very challenging for a user interactive application. User usually cannot manage such a great deal of vertices.

For object segmentation, we assume that we only need to manipulate segment size at least close to some common small objects in indoor scenes such as cup, mouse, mobile phones, shoes, etc. With this assumption, we design a bottom-up segmentation system. We first apply automatic segmentation algorithms to generate supervertices and regions. Users can then manipulate supervertices and regions with a set of simple operations to segment objects.

Graph-based segmentation

- Geometric segmentation on mesh vertices.
- Supervertex is the smallest geometric unit to manipulate.
- Each scene has ~5000 supervertices.

P. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *IJCV 2004*.

The first level in our bottom-up segmentation is designed for fine-grained manipulation.

We run a graph-based segmentation to group mesh vertices into small clusters, or supervertices. This is the minimal segmentation unit that a user can manipulate. Parameters of graph-based segmentation can be adjusted to change the size of a supervertex, usually based on the size of some common small objects in the scene.

In our experiment, we reduce millions of vertices to thousands of supervertices. In the SceneNN dataset, on average we have 5000 supervertices per scene.

Markov random field

- Geometric + color segmentation on mesh vertices.
- Attempt to group similar supervertices together.
- Each scene has ~500 regions.

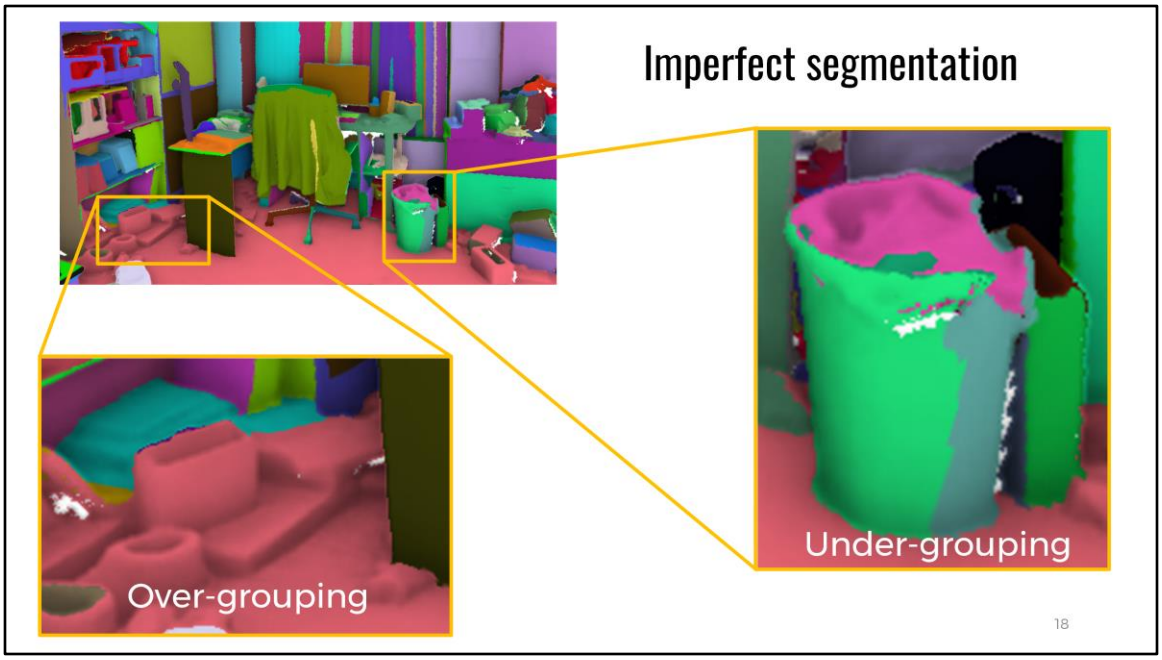
17

Next, we build the 2nd level of bottom-up segmentation by grouping the supervertices together using geometry and color information. We formulate this problem using a Markov random field. The output of the optimization produces a few hundred regions, which are more manageable for the user to refine.

Typically, the number of regions is a order of magnitude less than the number of supervertices. In SceneNN we usually have a few hundreds of regions.

The results of automatic segmentation are then presented to user for refinement. However, automatic segmentation is never perfect, and we would like to assist users as much as we can to refine the imperfect segmentation.

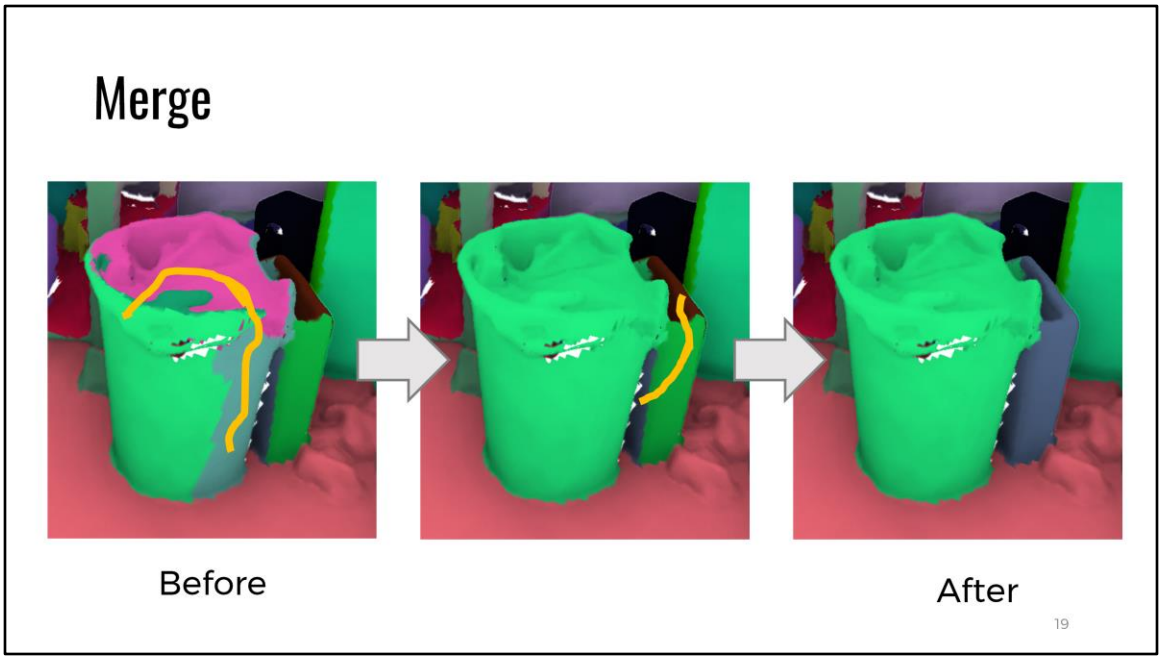
(In fact, a graphcut algorithm could be used here, but using MRF provides the capability for more sophisticated cost terms such as object probability, co-occurrence probability when grouping the supervertices. This could be a future research direction.)



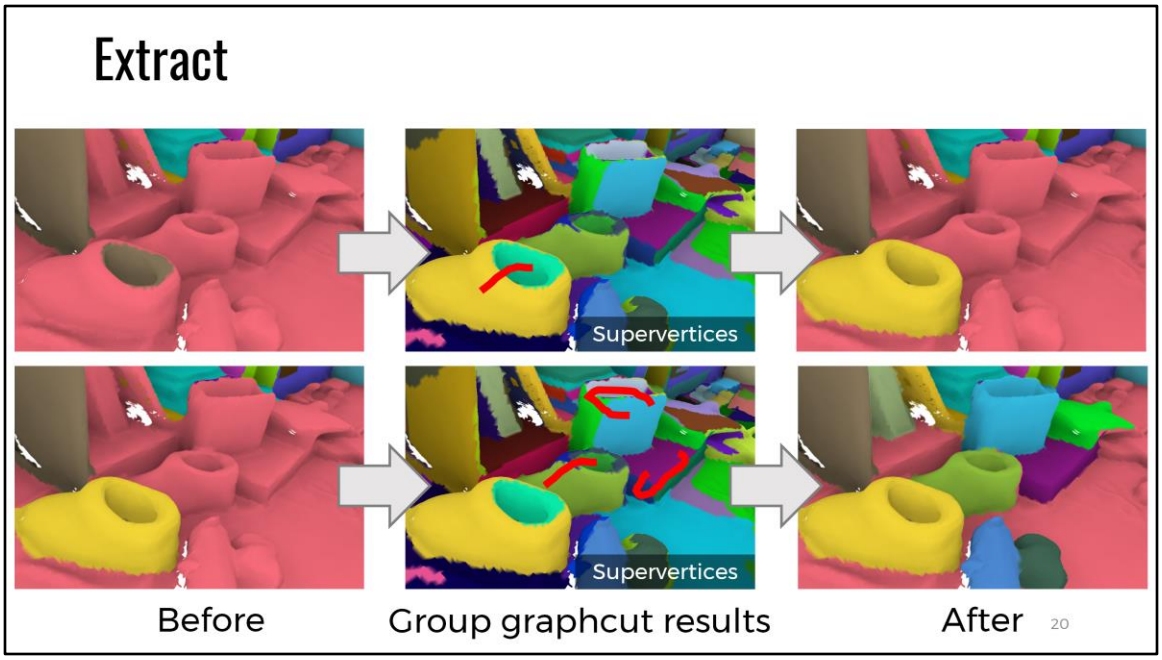
From the perspective of bottom-up segmentation, there are two types of imperfections: over grouping, and under grouping.

In over-grouping, the segments are too small and only form partial objects. By contrast, in under-grouping, the segments are too large and spread over many objects.

We address this problem by designing a set of simple operations for users to refine the segmentation.



Merge is the most common operation in our system and is used to solve over-segmentation. It merges several small segments to form a bigger segment which represents a semantic object in the scene.

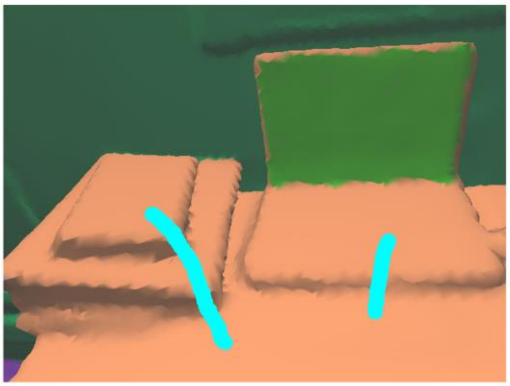


Extract could be seen as merge applied on supervertices for retrieving small segments from existing large segments.

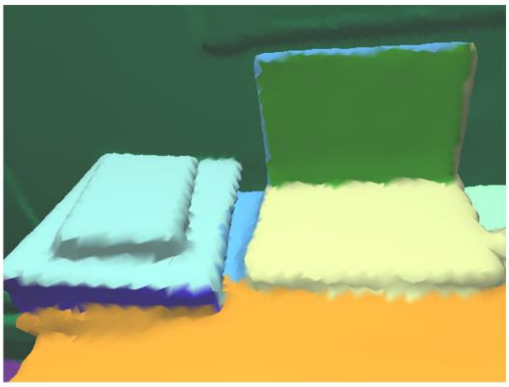
In this example, the first and third column show the current segmentation result. The middle column shows supervertices.
When user draws a stroke, supervertices are merged, and then the new regions are written to the current segmentation.

In our system, graphcut result is displayed on the fly when user press Alt key. Supervertices are precomputed. Therefore, merge and extract could be used without any significant overhead.

Split



Before



After

21

Split operation is designed to when the MRF-based segmentation need to be customized on specific cases. It performs the MRF-based segmentation on a single large segment, to break it into several smaller segments.

We also support undo in the system.

With this basic set of operations, user can now segment objects in a scene.

User interaction

- Simple operations: merge, extract, split, undo.
- Cache graph optimization results for fast switch between current regions and super-vertices.
- ~15 – 30 mins for a typical 16sqm room.

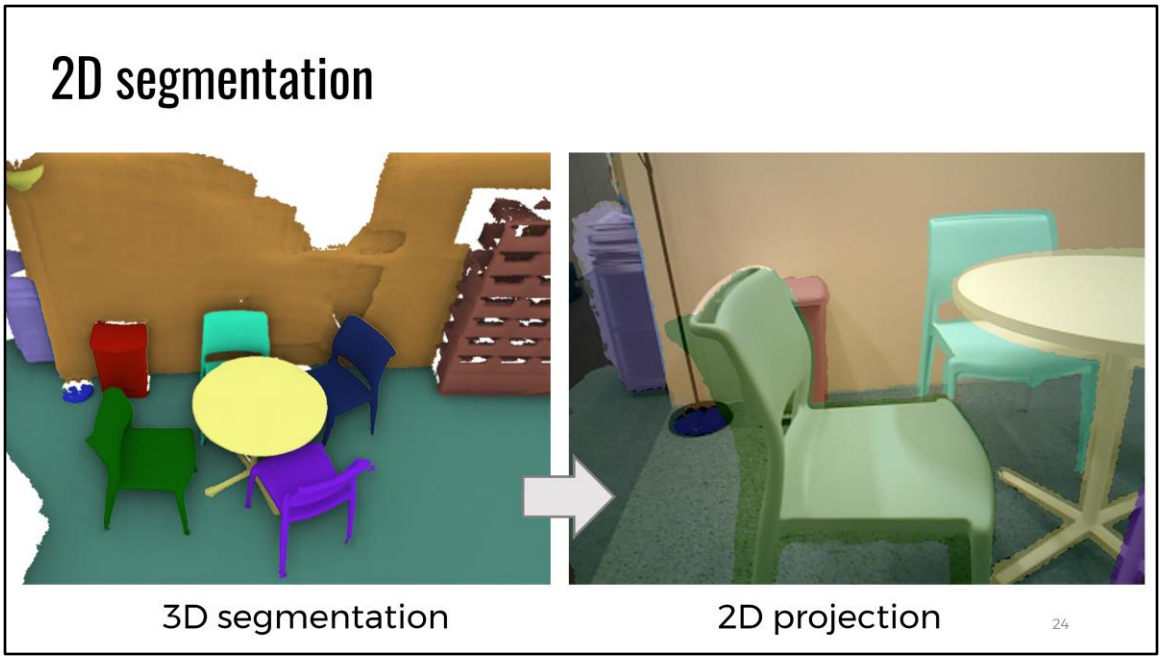
22

Here is an example that demonstrates annotation in progress.

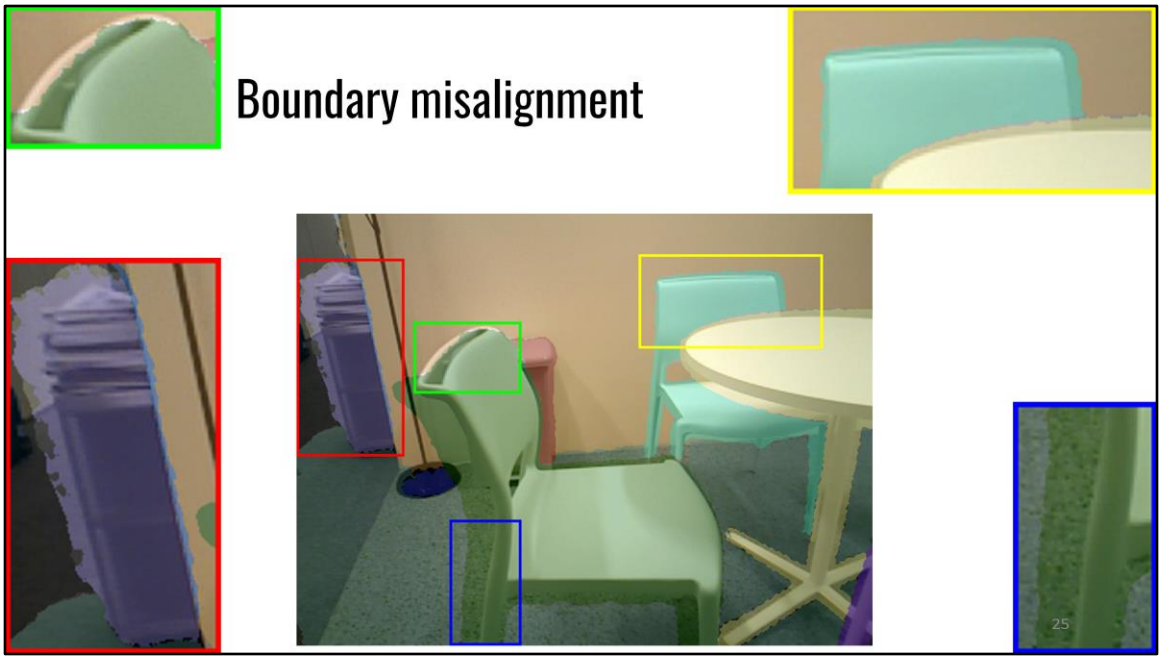
Advanced features

23

Beside a set of core user operations, we also investigate and design advanced features to assist users to obtain more accurate segmentation in a shorter time. Here we present two problems and their solutions: refining 2D segmentations, and handling repetitive objects in a scene with template-based object search.

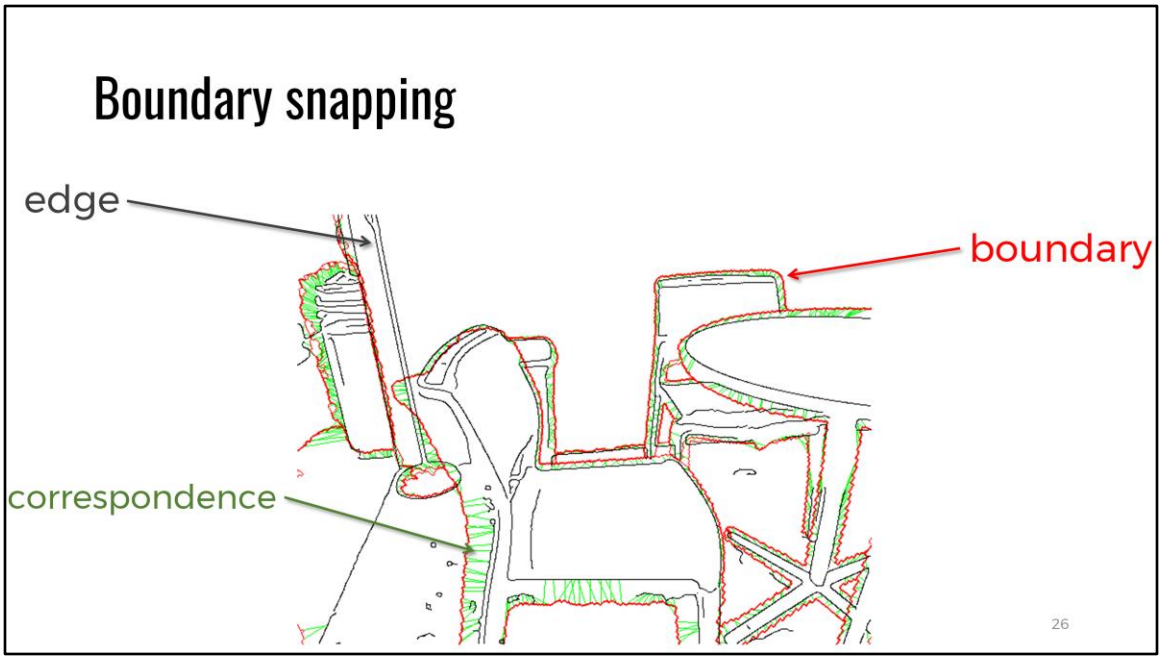


In order to obtain segmentation for 2D frames, we reproject the 3D segmentation results onto the 2D frames using the camera poses obtained from 3D reconstruction.



While the surfaces are aligned on the 2D frames, the boundaries can be misaligned.

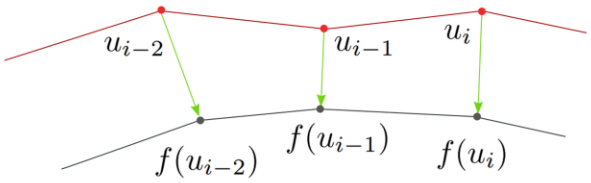
This is due to several reasons: inaccurate camera pose estimation or original color/depth images in RGB-D video has some misalignments.



To resolve this problem, we propose a general method to snap the boundaries of the 2D segments to the edges in the color images. This is formulated as a contour matching and dynamic programming.

Here the red contours are from segmentation image. The black edges are from Canny detector applied on color images.
Our task is to find a optimal set of correspondences to snap the red contours to the black edges.

Boundary snapping



- Find correspondences such that

$$\underset{f}{\text{minimize}}\left[\sum_{i=1}^{|U|}\chi^2(h_{u_i}, h_{f(u_i)})\right]$$

Difference of histogram of orientations

Continuity prior $+\kappa_1\sum_{i=2}^{|U|}\|f(u_i)-f(u_{i-1})\|$

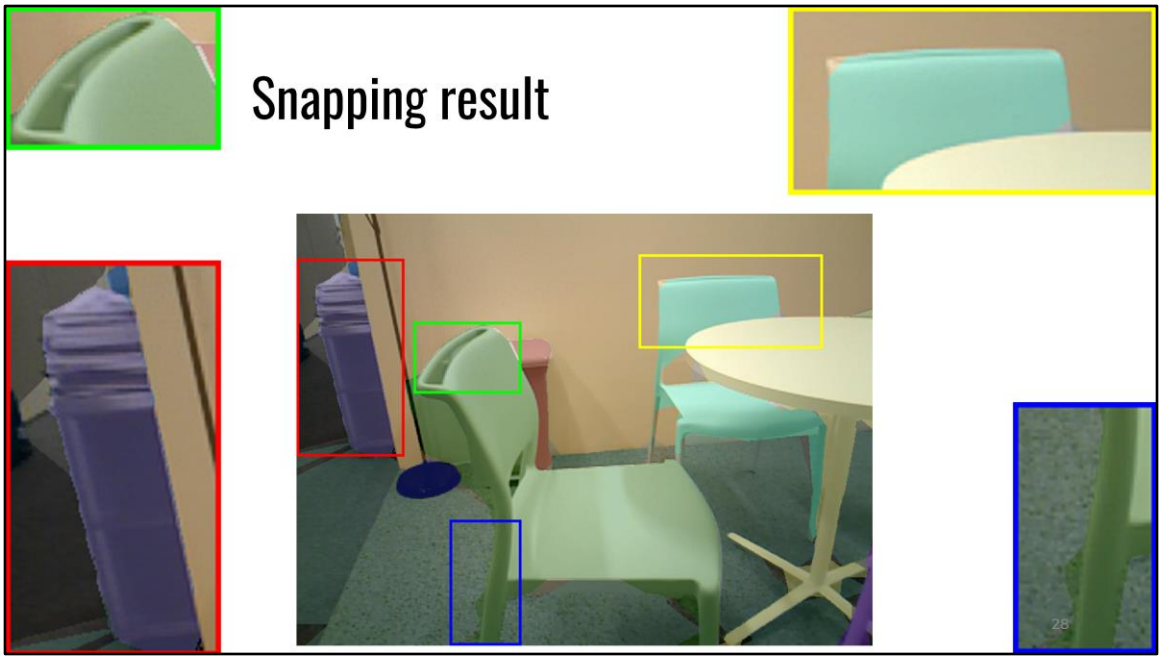
Smoothness prior $+\kappa_2\sum_{i=3}^{|U|}\cos(f(u_i)-f(u_{i-1}), f(u_{i-2})-f(u_{i-1}))\Big]$

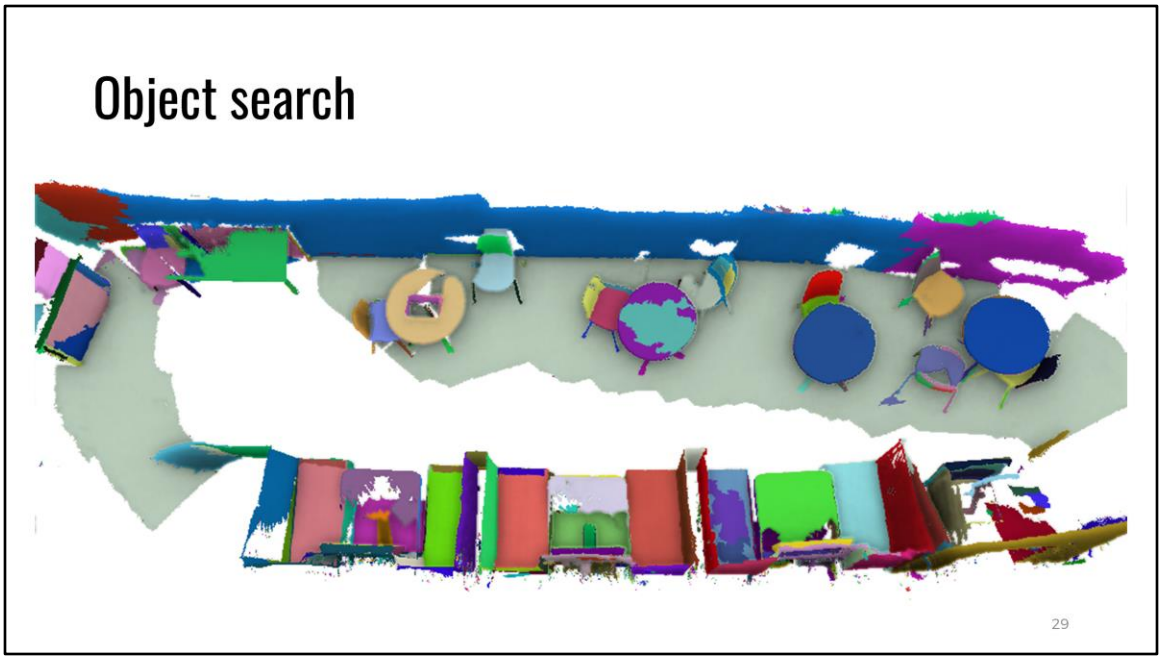
Optimization details in the paper

h is the histogram of orientations, or basically the descriptor at each point.
u is a point in the pointset.
f is the correspondence function that maps a point in the segmentation point set to the color point set.
X^2 is the distance function between two descriptors (see the paper, equation 5).

Find f such that the cost function is minimize:

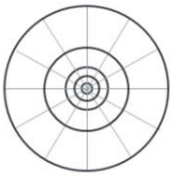
Continuity prior: the mapped points should be near each other.
Smoothness prior: the points after mapping should have low gradients.


Snapping result
28

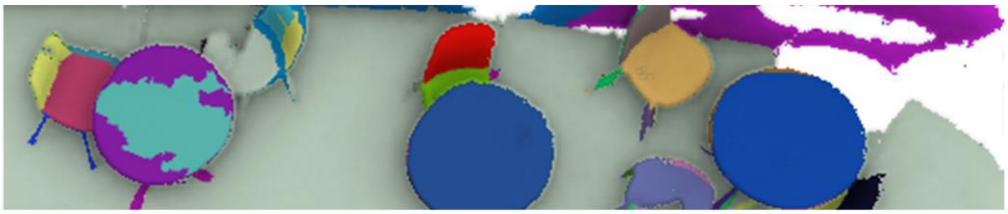


Our tool also allows the user to quickly annotate repetitive objects. Here is the process.

Template-based object search



- 3D shape context descriptor
- Sliding window search
- For each candidate region:
  Apply a greedy grow-shrink procedure to find the best combination of labels



30

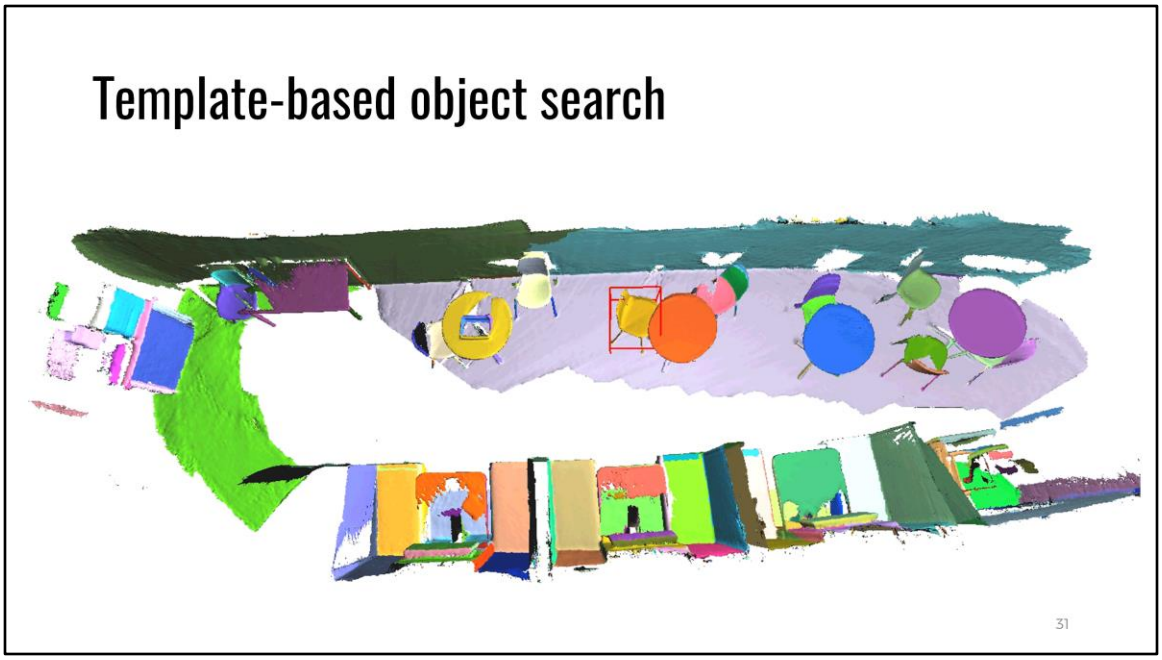
The user first chooses an example object. Our tool will then automatically search for similar objects in the scene.

We build template-based object detector using sliding window search. For each region, we generate the candidate object by apply a grow-shrink procedure. Grow means the current region will accumulate its neighbour regions, and shrink means it will discard a region from its current region set.

Each time grow or shrink is applied, we compute the 3D shape context descriptor of the candidate, and attempt to match its descriptor with that of the template. To improve matching accuracy, we also estimate a rigid transformation between the candidate and the template.

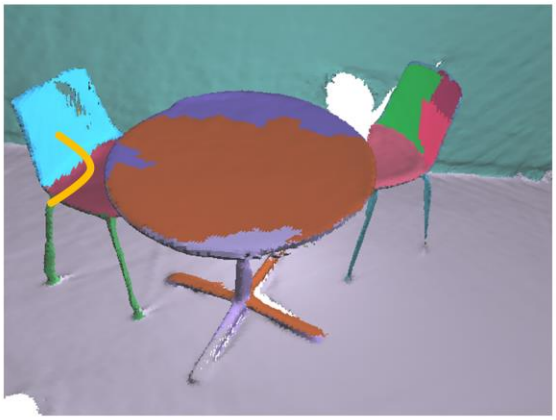
Matched objects in the scene are automatically highlighted for quick annotation.

Here is an example.



In this example, the chair enclosed by the red box is selected as the example object. It then performs the search in a sliding window fashion, with a grow-shrink procedure to find good matches.

Guided merge

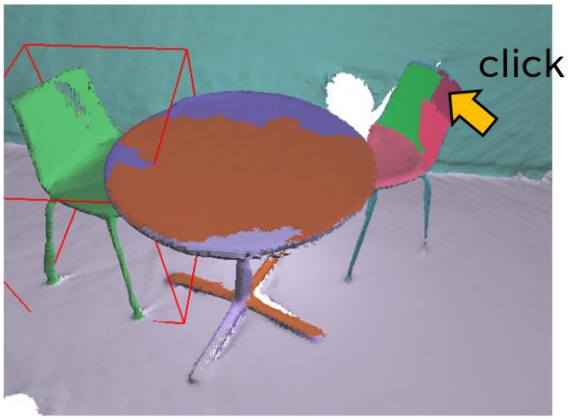


Apply grow-shrink after user interaction

32

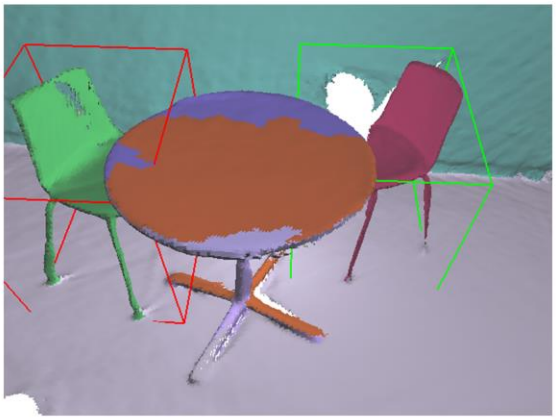
To handle objects that cannot be found by sliding-window object search, we introduce guided merge, a semi-automatic way to merge regions of an object. User first selects a template, and then click on a region of the target object. The grow-shrink procedure in object search will automatically find a set of regions near the click location that appear the most similar to the template. In this example, the chair on the left is used as the template, and a dark pink region of the chair on the right is clicked. The final merge is displayed in the green box.

Guided merge



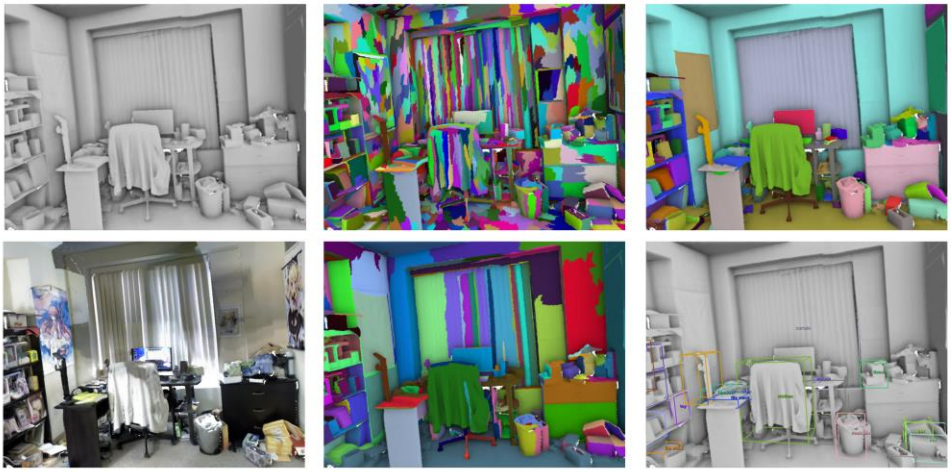
33

Guided merge



Experiments

SceneNN dataset annotation



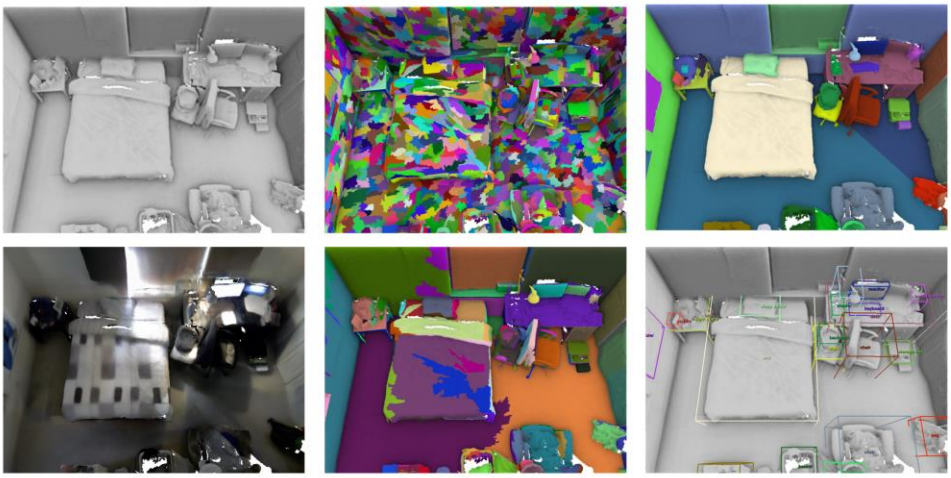
Reconstruction | Automatic segmentation | Refined segmentation

36

First we demonstrate the automatic segmentation and refined segmentation of a few scenes.

From a bedroom…

SceneNN dataset annotation



Reconstruction | Automatic segmentation | Refined segmentation

37

SceneNN dataset annotation

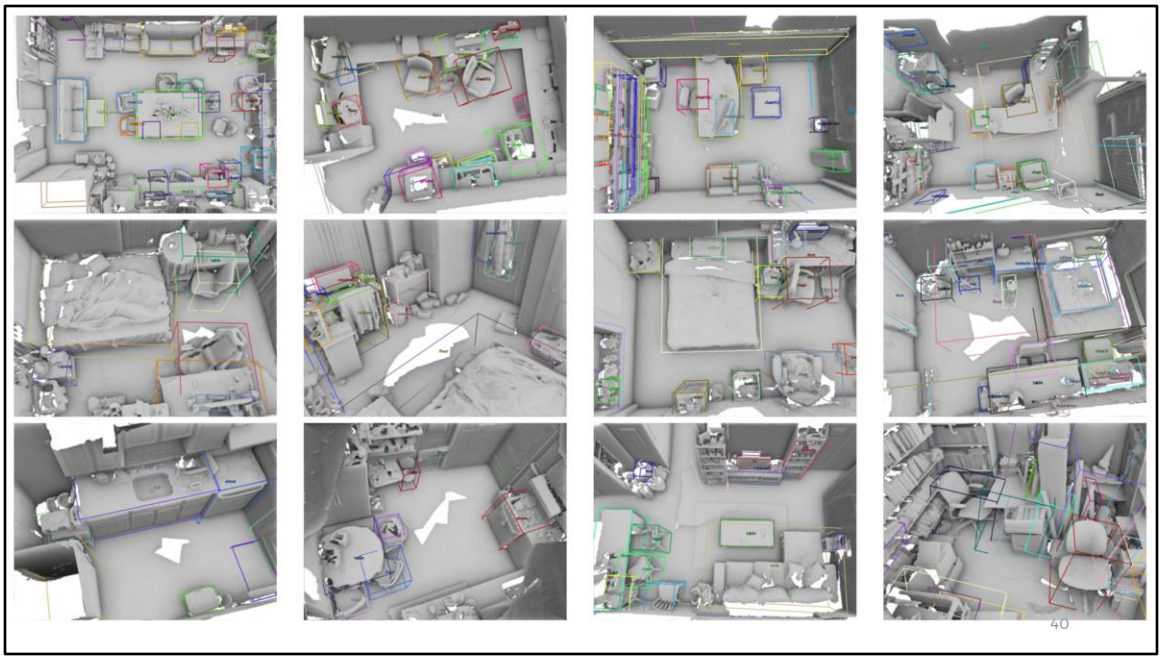
Reconstruction | Automatic segmentation | Refined segmentation

38

To a meeting room…

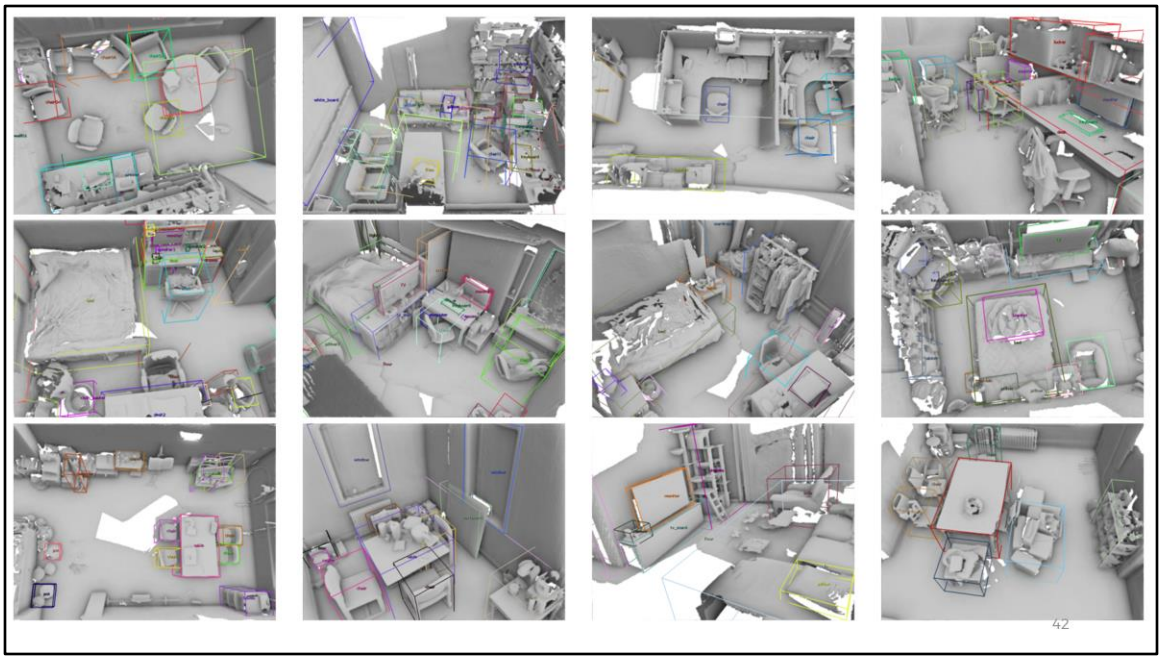
Note that with the same graphcut parameters, if we have more vertices, basically we will have more supervertices.

Here are more examples.



We also show the annotated bounding boxes with labels.


41


42


43

Outdoor Scene Annotation

45

Our tool is very general, and it can handle both indoor and outdoor scenes. Here is an example.

We are working towards improving the automatic segmentation with data collected from the tool itself. Gradually with more training data, the automatic segmentation will get better and better.

Automatic segmentation statistics

Scene	#Vertices	Graph-based			MRF-based		
		#Supervertices	OCE	Time (seconds)	#Regions	OCE	Time (seconds)
copyroom	1,309,421	1,996	0.92	1.0	347	0.73	10.9
lounge	1,597,553	2,554	0.97	1.1	506	0.93	7.3
hotel	3,572,776	13,839	0.98	2.7	1433	0.88	17.8
dorm	1,823,483	3,276	0.97	1.2	363	0.78	7.8
kitchen	2,557,593	4,640	0.97	1.8	470	0.85	12.2
office	2,349,679	4,026	0.97	1.7	422	0.84	10.9
Our scenes	1,450,748	2,498	0.93	1.4	481	0.77	12.1

46

And here is the quantitative experiment. We first show the automatic segmentation.

OCE measures the consistency of vertex labels w.r.t. ground truth. The lower OCE the better.

Here we use our interaction to first make the ground truth, and then use it to evaluate the automatic segmentation. As can be seen, MRF segmentation has better consistency.

The number of vertices is also reduced to thousands, and the number of regions to hundreds.

User interaction statistics

Scene	#Vertices	User refined #Labels	User refined #Objects	Interactive time (minutes)
copyroom	1,309,421	157	15	19
lounge	1,597,553	53	12	16
hotel	3,572,776	96	21	27
dorm	1,823,483	75	10	15
kitchen	2,557,593	75	24	23
office	2,349,679	69	19	24
Our scenes	1,450,748	179	19	30

47

Here is the statistics of user interaction phase. On average we annotate about 10 to 20 objects per scene in up to 30 minutes.

We target not only objects, but also clean separation of the objects.

We avoid label spilling (inaccurate label spread from one object to another) as much as possible.

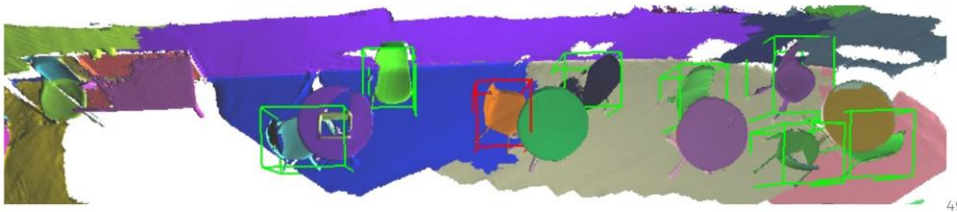
We also provide a qualitative comparison to the SemanticPaint system. Note that here we only compare system to system. We label the objects using categories defined in NYU dataset. There are a total of 37 categories.

As can be seen, our interactive system only results in low-frequency noise as we work on supervertices.

By contrast, SemanticPaint's conditional random field is applied to propagate label for each vertex, resulting in noisier labelling even on a flat surface.

Object search evaluation

- 45 objects in two categories, chair and table.
- Each object is used as a template.
- 69% precision and 70% recall.
- Template represented by 150 points.
- Search completes within 15 seconds.



49

We performed an evaluation of the object search performance. Since our object search is designed to handle repetitive objects, we only select scenes with such property in the dataset for evaluation.

With two common categories like chair and table, we obtain ~70% precision and recall, which proved that this technique could be useful when a lot of repetitive objects are in the scene, e.g., a class room scene.

Here we also tune the performance so that the object search can complete at interactive rate. In our experiment, it completes from 10 to 15 seconds.

Boundary snapping evaluation

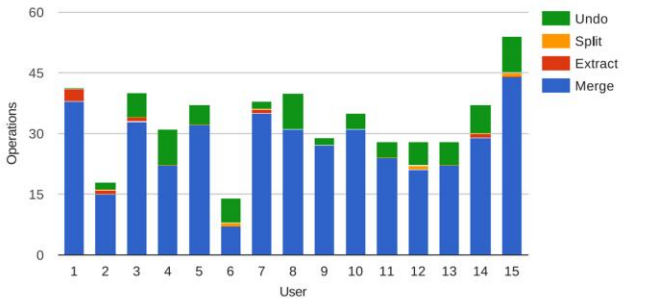
Segmentation method	OCE
Projection	0.57
Local shape	0.60
Local shape + Continuity	0.55
Local shape + Smoothness	0.55
Local shape + Continuity + Smoothness	**0.54**

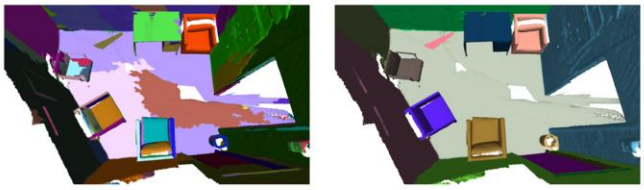
50

We also provide a quantitative measure of our boundary snapping technique. Again, OCE measure shows that the consistency is improved after boundary snapping is applied. This experiment also shows the effectiveness of the proposed priors.

User study

- To measure how merge and extract are used in practice.
- Task A: Simple scene, 2 minutes.
- Merge is dominant.





(a) Task A (two minutes)

51

Finally, we have conducted a user study on the effectiveness of our interactive operations (merge, split, extract, and undo).

We recruit 15 human subjects. Each participant is asked to perform two tasks (A and B) designed for simple and complex scenes. In task A, users were asked to segment a scene with only a few chairs and a table in two minutes.

In task B, users were required to segment a complex bedroom scene containing a significant amount of furniture and many small objects in ten minutes.

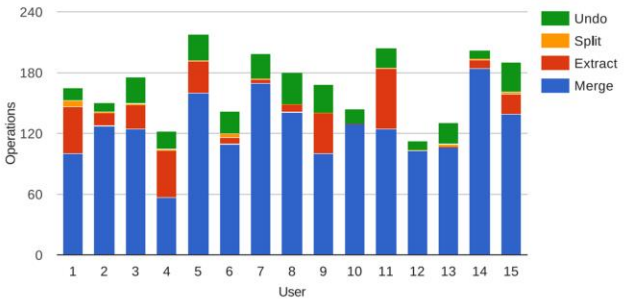
All operations users performed were logged.

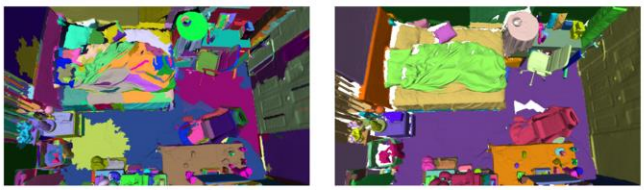
As a bottom-up segmentation, it is not surprising to see that merge operation is dominant. In a simple scene as in Task A, there is little over-grouping. Therefore, extract and split are rarely used. There are some undos as first-time users are not familiar with the tool.

User study

- Task B:
  Complex scene,
  10 minutes.

- More extracts used
  in complex scenes.





(b) Task B (ten minutes)

52

In a more complex scene, we can see that extract becomes more substantial as over-grouping now appears more due to more small objects appear.

In fact, we also learn that, the definition of objects could be quite different from the perspective of each human subject.
For example, for a table, some users favour segmenting the table top of an object, while others group the table top and table legs together.

For both tasks, our subjects are able to segment the objects using the proposed operations.

WebGL annotation tool

http://scenenn.net/webgl/index.html

- Reimplementation of our original C++ annotation tool.
- Graphcut, MRF with merge and extract.
- Open source.

53

At start, we implemented our tool in C++ and used it to annotate more than 100 scenes.

To support more scalability, we now have the tool reimplemented in WebGL. Basic features such as merge, extract, undo, graphcut, MRF are supported.
You can now navigate to www.scenenn.net/webgl to try.
You can also inspect the source code and modify it for your own use.

We will open a new GitHub repository to officially host the WebGL source code soon.

WebGL annotation tool

http://scenenn.net/webgl/index.html

At start, we implemented our tool in C++ and used it to annotate more than 100 scenes.

As a part to test scalability, we now have the tool reimplemented in WebGL. Basic features such as merge, extract, undo, graphcut, MRF are supported.
You can now navigate to http://webgl.scenenn.net to try.
You can also inspect the source code and modify it for your own use.

Future work

- Less time, more quality.
  Initial segmentation powered by a deep network.
- Better user experience.
  Online learning of user operations to reduce undo.
- Annotate more scenes for 3D deep learning.

55

There are many research directions from the current work.

First, we can strive for better speed and quality. With the recent advance of deep learning, we have thought of integrating neural networks to improve the quality of automatic segmentation. This is a work in progress.

We also would like to improve the user experience. For example, after some merge and extract, the system could be able to learn the pattern of this user, and thus suggest object segmentations according to previous user inputs. This form of online learning would help to adapt the tool better because as we said, the definition of objects by each user could be very different.

Finally, we would like to scale the annotation to more and more scenes for deep learning use.