Real time spatio temporal segmentation of RGBD cloud and applications

by

Amritpal Singh Saini

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

There is considerable research work going on segmentation of RGB-D clouds due its applications in tasks like scene understanding, robotics etc. The availability of inexpensive and easy to use RGB-D cameras and computational capabilities of GPUs has lead to development of numerous applications in this area. Recently proposed cloud segmentation methods are either slow in operation or do not operate in an online fashion making them unsuitable for applications in robotics. In this work we deal with the aforementioned problem. We propose a method to perform online segmentation of RGB-D scene. Our framework is built on dense scene mapping methods like Kinect fusion. It allows us to generate accurate and dense depth maps and provide camera pose information. Instead of directly operating on a large 3D point cloud we process individual RGB and depth frames which are assembled in a dense cloud in an incremental fashion. Pose information is used to integrate the segmentation maps into the global label cloud using GPU. We perform multi-view integration of segments as the camera is moved around in the scene by formulating the problem as weighted graph. We will discuss applications of our segmentation framework to perform real time and scalable object discovery and object detection.

Acknowledgements

I would like to thank my supervisors Dr. Nilanjan Ray and Dr. Hong Zhang. During the period of my graduate studies at computing science department at University of Alberta they not only provided me with learning opportunities at various projects but also provided immense amount of support and feedback on the thesis project. Thank you for being so helpful and understanding during the period of my studies. Working with you have a great learning experience and I have not only learned technical concepts from you guys but also learned a lot of about work ethics. This knowledge will help me a long way in life. I also have to thank Dr. Judith Hugh and Dr. Russ Greiner, I gained a lot of practical experience while working on breast cancer problem and that experience helped me a lot in my professional career. I really appreciate my course instructors Dr. Martin Jagersand, Dr. Pierre Boulanger, Dr. Dale Schuurmans who helped me gain the understanding of basics of various areas like computer vision, medical imaging and machine learning.

I also have to thank everyone that I met at University of Alberta who provided me support either directly or indirectly during the period of my studies. Arnamoy Bhattacharyya, Ryan Kiros, Alejandro Ramirez, Kyrylo Shegeda, Andy Hess, Usman Aziz, Mohammed Waqar, Rajarshi Maiti, Camilo and Carol Perez thanks for being an amazing company, I learned a lot from you guys. Also thanks to Garima Mehta for all the fun experiences with cooking different varieties of dishes, I almost gained weight. This list will be incomplete without Sonya Sehgal who I have to thank for all the stimulating discussions about various topics besides computer science.

Thank you mom, dad, little sister Indu and Grandpa for being so awesome and patient and mailing me packages of home made food. Thank you for

supporting all my endeavours and decisions.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Vision is one of the primary senses used by us to interact and understand the environment around us. We can learn to recognize large number of objects in various poses and under different lighting conditions with great accuracy. We can not only learn the appearance of various objects but also reason about the interaction between various objects, just from visual stimuli. Owing to the detailed information that visual stimuli can provide us, there has been considerable interest to understand the workings of visual cortex and have machines perform similar task.

Such a system will have numerous application in industrial automation, robotics, medicine etc. However besides some successes in tasks like face recognition by [28], [40], text detection and optical character recognition by [15] there have been lack of a general solution that can perform as good as human visual system. The primary reason for lack of progress in this area has been due to the lack of suitable computational hardware and associated cost, and the lack of understanding of working of visual cortex and nervous system.

1.1 Motivation

Human visual system process various low level cues combined with high level reasoning to group and classify each element of visual stimuli belonging to certain object class. Intensity and color information provides a large amount of information about the scene. It is one of the basic cues which help us to delineate object boundaries. But only intensity is not sufficient to completely

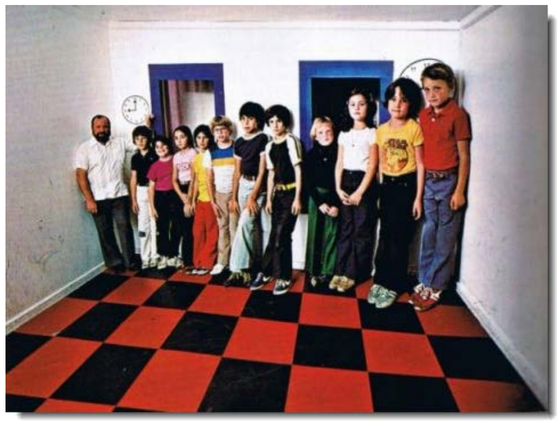


Figure 1.1: Ames room experiment. Absence of other low level cues besides intensity distorts our perception of a given scene. Source: http://www.eyes-and-vision.com/optical-illusion-ames.html

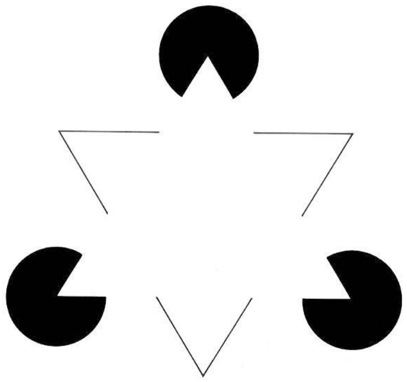


Figure 1.2: Kanizsas triangle illusion, we automatically fills in the white space with the image of a triangle. Hence, we perceives the image as consisting of two triangles. Source: http://thetartan.org/2009/2/9/scitech/howthingswork

understand a given scene. For example in the famous Ames room experiment an image of a distorted room is captured which demonstrates how we reason about the objects in the scene when depth or other low level cues are absent. See figure 1.1

Depth, motion, texture, shading and shadow interact in a complex manner with color to further provide information about the scene. We demonstrate the complexity of the visual system in the following figures 1.2 1.3 1.4 1.5 .

Last but not the least, context and culture also plays a very important role in the task of scene understanding. An interesting example of role of context is

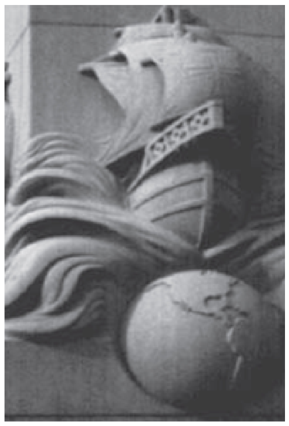


Figure 1.3: Highlights and shadows help us perceive depth. Source: http://what-when-how.com/computer-vision-from-surfaces-to-3d-objects/scene-statistics-and-3d-surface-perception-computer-vision-part-2/

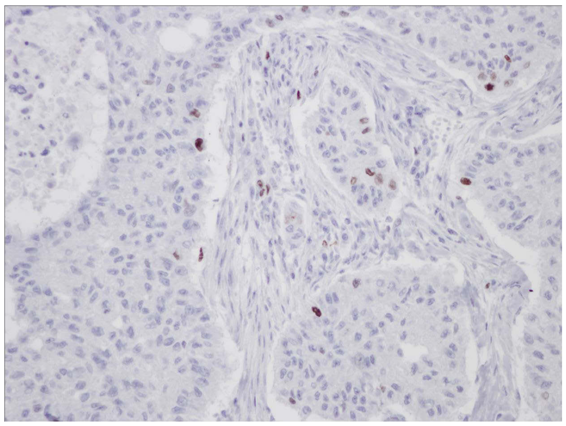


Figure 1.4: Combination of intensity and shape are used to find nuclei in an image.

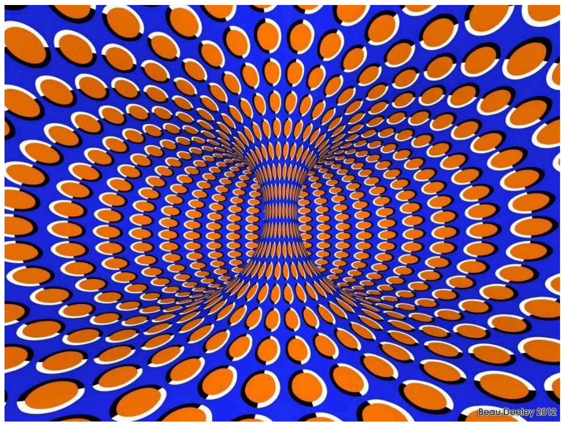


Figure 1.5: Combination of disconnected components arranged in a specific pattern forms illusion of depth. Source: http://monkeybuddha.blogspot.ca/2013/01/the-optical-illusion-of-dimension.html

shown in the image figure 1.6. The way the image is perceived and understood depends on culture the observer is from.

Another low level cue of interest is motion. Motion has been used in many important research works for tasks like novel object detection, object tracking, depth from motion. In our day to day interaction we can reason about the objects and their components based on motion.

Aforementioned examples demonstrate how various low level cues combine with each other to provide information about the scene. But the manner by which all these cues are combined and processed is not completely understood. Physiology of human eye also plays an important role in interpreting the visual stimuli. It has been shown that rather than processing every part of the scene we just focus on some important aspects like higher relative contrast, motion, depth. Also most of the processing of visual stimuli is performed in the area of retina known as *fovea*. Due to the complexity of the system, it has lead to the development of various research problems in computer vision.

Object localization and recognition is one such problem which deals with localization and recognition of objects. Some examples of prominent works in this direction are by [10], [43], [27]. General version of object recognition problem is known as scene understanding [26]. The problem of scene understanding is closest to capabilities of human visual system. It deals with assigning every

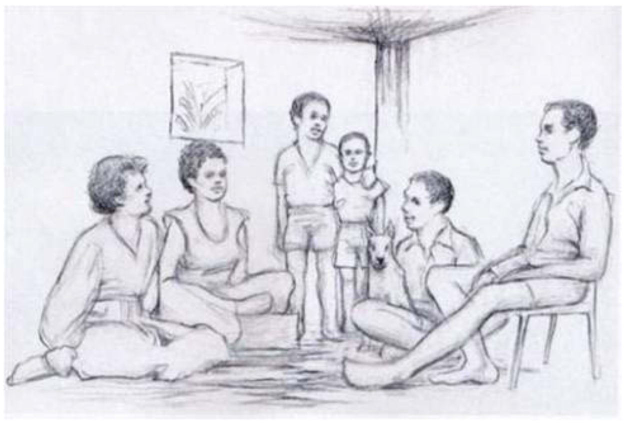


Figure 1.6: When the image is shown to people from East Africa, nearly all the participants in the experiment said she was balancing a box or metal can on her head. In a culture containing few angular visual cues, the family is seen sitting under a tree. Westerners, on the other hand, are accustomed to the corners and boxlike shapes of architecture. They are more likely to place the family indoors and to interpret the rectangle above the woman's head as a window through which shrubbery can be seen.

pixel its object class, which leads to solving the problem of object localization, recognition, and segmentation simultaneously. Scene understanding is one of the most active research area and with the challenges like PascalVOC challenge [11] and databases like Imagenet, novel solutions are being developed in this area. However this problem has not been solved with good accuracy, as it is computationally time consuming, does not scale very well to large number of object classes and depends heavily on large amount of training data.

Taking inspiration from the structure of human eye and attention mechanism has led to the development of another research area known as active vision. Active vision deals with detection and segmentation of objects in a scene where processing is done on a very small region of the image. This approach has an advantage of being very similar to human visual system and can process regions of interest in small amount of time. Recently there has been considerable interest in this area and some of the prominent works has been [29], [4].

In more recent works tasks like object detection are being performed by combining various cues like intensity and depth. Depth plays an important role for delineating the object boundaries and reasoning about a scene. With advances to problems like stereovision and development of sensors like Microsoft Kinect has made it possible to acquire depth information with good accuracy in a real time fashion. RGB-D scene segmentation, object discovery, and object recognition are some of the basic requirements in robotics and computer vision applications. For real world applications these problems should also meet the criteria of being real-time and online in order to operate in a dynamic environment. Currently there is no method available that can achieve these objectives *simultaneously*. In this work we develop a framework for online RGB-D scene segmentation which operates in real time, and discuss its application in problems like object detection and object discovery.

In recent years considerable amount of research has been performed on scene understanding by [26] [25], large scale object learning, and object discovery and detection by [23] [4]. These research problems have profound impact on the field of robotics. Solving these problems allows machines to interact, manipulate and reason about the environment. However the methods available are too slow in operation for applications in robotics.

More recently, depth is being used as an additional cue to perform scene understanding. Inexpensive RGB-D cameras like Microsoft Kinect have made it easy to acquire synchronized depth information with RGB information and this has led to new developments in scene understanding. Some of the prominent works in this direction are by [34] [7]. [23] has shown that objects can be segmented from a given mesh using learned geometric properties.

Performing object detection and scene labeling in an indoor environment is challenging. Presence of a large number of objects makes these tasks even more difficult. Thus, achieving scalability and real time performance has proven to be a difficult task. Existing approaches require a training database for every object that has to be recognized. E.g. [27] requires sliding window detector to be trained for every object. Creating a training database is a costly process. The other challenging aspect of the problem is adding new objects to be recognized and making the whole process online.

1.2 Overview

As discussed in the previous section human visual system group elements of visual stimuli using various low level cues. In this work we develop an online method for point cloud segmentation and demonstrate its application in object discovery. Nearest neighborhood methods proposed by [41] for object detection can be used to overcome the limitations discussed in the previous section. In this work we develop an online method for point cloud segmentation and show how object discovery and nearest neighborhood methods for object detection can be used to perform tasks similar to human visual system. Object discovery can be used to find new objects in a given scene. [19] used motion as a cue to detect novel objects in a given scene and [29] used depth data with color information to detect novel objects. For the methods using depth as cue rely upon segmentation of point cloud as prior for discovering objects for example [23] and [13]. But these segmentation methods operate on a dense point cloud in an offline fashion thus making them unsuitable for applications in robotics. It also affects the computational performance because as the size of point cloud increases so does the computational cost to segment it.

Nearest neighborhood based object detection methods using large databases overcome the limitations of training a classifier for every object. [41] used a modified sum of squared difference (SSD) as a metric to perform different tasks like object detection, image coloring using a large database on small images. In this work we use the same approach to perform object detection. As the camera is moved around, we leverage upon our multiview segmentation framework with object discovery to compute the large database in an unsupervised manner. We do image matching on GPU and use SSD as metric to compute nearest neighbors.

Main contributions of this work is a real time, online segmentation method for RGB-D data. Our proposed algorithm is built upon Kinect fusion large scale part of [35]. We perform computation on GPU with efficient segmentation methods to achieve run time performance of 5-6 fps for segmentation. We also discuss various applications of the segmentation framework. First application that we focus on is unsupervised or semi-supervised generation of object database. As discussed previously that collection of training data is a difficult process and requires human interaction. In this work we demonstrate how object discovery with point cloud segmentation can be used to generate object database. Second application of our framework is for scalable, real-time object detection. In literature different methods have been proposed to perform object detection for RGB-D data. However these methods invariably suffer from the problem of scalability. In this work we leverage computational power of GPU to perform nearest neighbor match for object detection based on work by [41]. We discuss application of proposed framework for object detection in an efficient and scalable manner using proposed online segmentation framework.

Chapter 2

Literature review

This chapter reviews the RGB-D cloud segmentation literature and various applications. We also define the problem statement and significance of our work. There has been extensive previous work on RGB-D scene segmentation and object detection. RGB-D cloud segmentation methods can be divided into two main categories. Some contemporary approaches use prior information for segmentation for example trained classifiers or sliding window detectors. Alternative methods use unsupervised segmentation by imposing various smoothness constraints on the cloud. As a background to the proposed method we also discuss dense point cloud reconstruction, object recognition using nearest neighbor and object discovery methods in the following sections.

2.1 RGB-D segmentation

2.1.1 Segmentation using prior information

Some of the common approaches use prior information to segment and label the given scene. This problem is also known as scene understanding. Features and object representations are learned from a labeled object data set and the learned models are applied in testing phase to perform segmentation and labeling of a given scene. [34] uses kernel descriptors proposed by [6] aggregated over superpixels and transformed using efficient match kernels used by [8] for RGBD scene labeling. Kernel descriptors were used for capturing shape similarities and appearance. For contextual modeling they have used two approaches, one using MRFs and other using segmentation trees constructed by using gPb contour detector by [3]. In contrast to their approach we use an unsupervised segmentation method and rely on temporal integration of segments to perform scene segmentation. For labeling we use nearest neighbor method and object discovery.

[27] used of sliding window detectors combined with inference on MRF over the voxels to produce scene labeling. They have obtained very good results for the task of scene labeling. However one shortcoming of this approach is that a sliding window detector has to be trained for every object making it difficult to scale the method to large number of objects. [46] proposed a modification of deformable parts model for RGB-D data to perform object detection in RGB-D scenes, however the method is not scalable and require offline training for object models. [2] also performs 6DOF pose estimation of objects besides recognition of the objects by combing features like SIFT for RGB data and SHOT for depth data with hypothesis verification. Using a contour based approach [16] modified gPb contour detector for segmentation of RGB-D scene. Besides RGB contour cues used in original gPb contour detector, they used three more geometric cues extracted from depth images to perform segmentation. After segmentation, various features like shape, geocentric pose are estimated from the superpixels and classification is performed to determine object classes. [39] focus on the problem of detecting objects in RGB-D by verifying the compatibility between object hypotheses and corresponding RGB-D map.

The techniques discussed in this section depends on training data which can be costly and time consuming to generate. They also suffer from the problem of scalability in terms of the number of objects. For example [27] requires a sliding window detector for every object detected. This technique does not scale very well to a large number of objects. An alternative to providing prior object information is using solutions developed for object discovery. These methods rely on either active vision [1] [29] or segmentation of scene and classification of segments as objects proposed by [23]. We will discuss use of object discovery to generate training data in the following sections.

2.1.2 Unsupervised segmentation methods

In contrast to the segmentation methods discussed in the previous section, most of the works in this section use smoothness or other geometric constraints to segment a given scene. [1] proposed an active vision framework using fixation to perform task like segmentation and object discovery. They perform segmentation using extension of gPb contours for RGB-D images by generating probabilistic boundary map. Then using a fixation point they generate polar representation of contour map and segment objects by finding ”optimal” closed contour. However gPb contour detectors are is slow in practice. Even a GPU implementation by [9] takes seconds to compute contours on an RGB image thus making it unsuitable for real time applications in robotics.

[33] performs segmentation of objects of interest using saliency maps and depth data captured from stereo. [24], [42] use motion and depth as a cue to perform segmentation of unknown objects. In their work objects are manipulated by a robotic arm to verify object hypothesis. In recent literature different segmentation methods have been proposed which are very fast in practice. [30], [23], [13] used convexity based criteria to segment the RGB-D scene. [23] operates on point cloud data obtained from Kinect fusion and extract multiple segments using [12]. For each segment, features are extracted and segments are classified as object. [13] uses a similar framework using free space filtering to find differences between two point clouds to learn about new objects. In proposed work we use similar segmentation criteria with slight modification to perform segmentation as explained later in this thesis.

[14] perform min cut segmentation of point cloud given object location. Even though the segmentation method is simple it requires object location. In contrast in our method we initialize object location using object discovery methods and instead of using multiple thresholds, we over segment the scene and rely on temporal segmentation for correct segmentation. [5] uses active vision for 3d scene segmentation of unknown objects. They create models for foreground, background, and object with stereo based fixation process to perform segmentation. With their proposed approach only one object can be

segmented at a time.

Another category of unsupervised object segmentation uses active vision approach inspired by human visual system. Humans can learn object representation in a purely unsupervised manner. Whereas current state of the art object detection methods rely heavily on a labeled datasets. The human visual system makes a series of fixations at various salient locations in a given scene [45]. The eyes move from location to location either voluntarily or involuntarily. The parts to focus on are selected based on various local cues. Local cues like relatively higher contrast, motion. Inspired from the behavior of human visual system, active vision selects some points in the scene to perform tasks like segmentation, object detection.

[1] has shown applications of active vision in various tasks for object segmentation, robotics. For object segmentation, using gpb contours an edge map is created. To segment the objects a point of interest is selected. The edge map is converted to polar map around the point of interest and by finding an optimal cut on edge map in polar space the boundary of the object is obtained.

Also [22] used an active vision based approach to segment 3d point clouds. Points of interest are selected by various methods such as geometric seeding, saliency based seeding and seed points selected by humans. An approximate solution to the segmentation is obtained by performing multiway cut as it is an NP hard problem. [37], [38] performs unsupervised analysis of images to build object class models.

Recently [20] proposed a temporal segmentation method which uses depth and color information. They segment the data using multistage hierarchichal graph based approach. Similar regions in several point clouds are grouped over a graph. These regions are merged to yield a dendogram using agglomerative clustering via minimum spanning tree algorithm. They achieved .8 fps on images of size 640X480.

Segmentation methods proposed by [23], [13] and discussed previously are either slow or operate in an offline manner thus rendering them unsuitable for applications in robotics. We refer to these offline segmentation methods as batch solutions as before performing segmentation they collect dense 3D

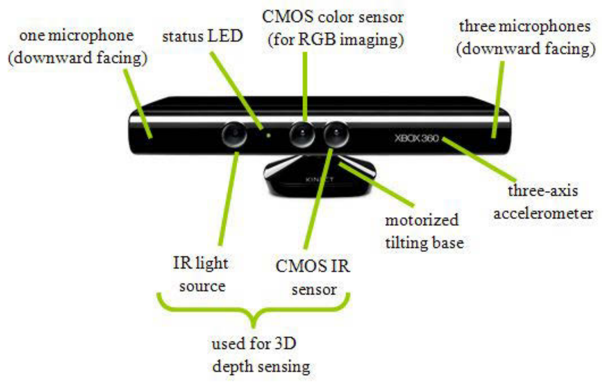

one microphone (downward facing)
status LED
CMOS color sensor (for RGB imaging)
three microphones (downward facing)
three-axis accelerometer
motorized tilting base
IR light source
CMOS IR sensor
used for 3D depth sensing

Figure 2.1: Microsoft Kinect. Source: http://fivedots.coe.psu.ac.th/ ad/jg/nui13/kinect.jpg

cloud of the scene. In contrast to these batch solutions we propose an online solution for the segmentation problem that is sequential i.e. processing one RGB-D frame at a time.

2.2 Dense reconstruction methods

As we have highlighted before, human visual system combine many low level cues to understand a scene and depth has proven to be one of the very important cues. This property of visual system has lead to development of problems like stereovision, depth from single camera, depth from motion. It has also lead development of some novel devices and sensors which project structured light which is used to infer depth in real time. Such low cost and easy to use devices have been used in many recent works in robotics and computer vision.

Microsoft Kinect is one such sensor 2.1. It has an infrared light source which projects structured light 2.2. Infrared camera and on board processing capability is used to determine depth from the projected light. Besides depth camera, it also have a RGB camera and provides registered stream of RGB and depth information. Besides cameras it also has other on board sensors like accelerometer and a motorized base, microphones. But Kinect also suffers from few limitations like missing data due absorption of infrared rays, shadows and noise. Black surfaces causes absorption of infrared rays causing loss of

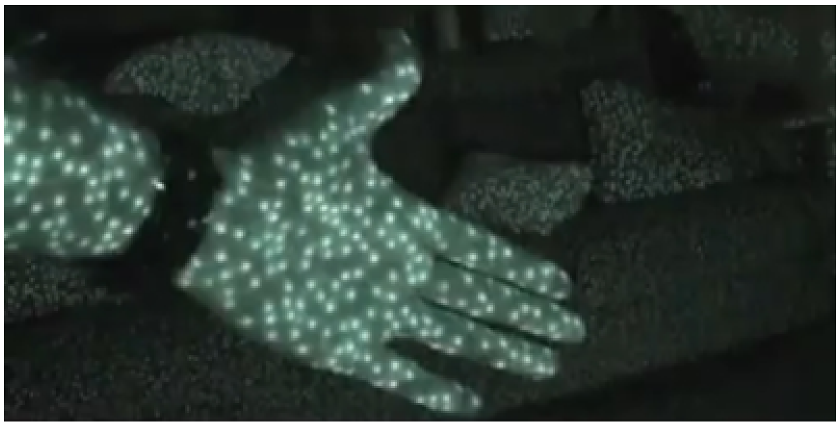


Figure 2.2: Structured light pattern projected by Kinect. Source: http://blogs.msdn.com/b/msroboticsstudio/archive/2011/11/29/kinect-for-robotics.aspx

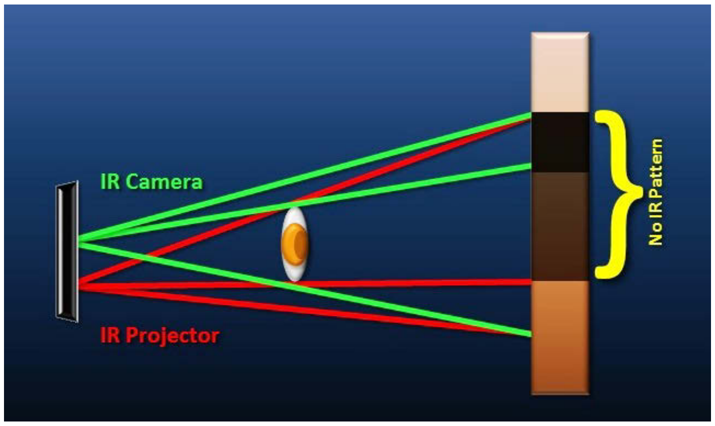


Figure 2.3: IR shadow Kinect sensor. Source: http://social.msdn.microsoft.com/Forums/en-US/74ff175a-291f-445d-ab55-09d2af7cfd4c/why-did-my-kinect-sensor-show-such-a-double-image

information. The phenomenon of shadows is highlighted in following figure 2.3.

One way to overcome these sensor limitations is by using dense reconstruction methods. Dense reconstruction allows to generate dense 3d point cloud of a given surface. Few prominent works in this area has been by [44]. Kinect fusion developed by Microsoft demonstrated the use of Kinect sensor to perform dense reconstruction and various other applications. Kinect fusion perform camera pose estimation using ICP and integrate raw depth maps into a global TSDF (Truncated Signed Distance Function) cloud. GPUs were used to obtain real time performance. The imprtant point to note is that GPUs played an important role in feasibility of the method. ICP for pose estimation works only when two point clouds to be aligned are very close to each other. GPU made the processing fast enough that ICP can be performed at much faster frame rate. Another important aspect of the kinect fusion work is that rather than doing frame by frame registration, authors have performed frame by model estimation, thus reducing the pose error.

But original kinect fusion suffered from the limitation of amount of area that can be mapped. The primary reason for his limitation is that GPU memory is limited and a costly resource, and data transfer between CPU-GPU involves considerable overhead. Later Kintinous was developed which allows mapping of large amount of area by swapping memory blocks representing a part of the surface in and out of the GPU. As the sensor is moving around is a transformation above certain threshold is detected current TSDF (Truncated Signed Distance Function) GPU block is swapped out of GPU memory and new part of the surface is mapped is the current cloud. This approach helped to overcome memory limitation of the GPU. Another noteworthy aspect of Kintious is pose estimation process. They have proposed FOVIS (Fast Odometry from VISion) [21] as pose estimation method which is more robust in comparison to ICP during large amount of camera transformation.

Our algorithm is built upon dense reconstruction algorithm similar to Kintinous. We work with an open source implementation known as Kinect fusion large scale implemented in Point Cloud Library [35]. The dense mapping algorithm provides our method with dense depth and pose estimates. Which are used for further processing like scene segmentation and multi-view integration of segments into a label cloud.

2.3 Nearest neighbor object detection methods

As discussed in the previous sections that there have been any methods proposed in literature that deals with object detection in RGB-D clouds. Most of these methods rely upon labeled datasets for training and use model based approaches for object detection. But generating ground truth data is not only costly but also time consuming and not scalable. Another limitation of these methods is that they perform object detection using model based approach i.e. for every object to be detected a prior model has to be trained. This severely reduce the scalability of object detectors in terms of number of objects that can be detected.

The other alternative to model based methods is model free methods for example nearest neighbor. Nearest neighbor methods use large database of objects of interest to perform various tasks like object detection, segmentation, image coloring etc. These methods usually have advantage of being highly scalable in number of objects that can be detected and do not require training of complex models for every object.

One of the prominent works in computer vision using nearest neighbor methods have been [41]. They used a database of 80 million images and demonstrated that using even images of seize 32X32, tasks like object detection, image coloring, object localization can be solved with good accuracy. Given a query image, they first prune candidates based on top 19 eigen vectors. After selecting a subset of whole database they compute nearest neighbor using modified SSD score which supports warping. Because of low resolution of the images these computations can be done easily. We use a similar non parametric approach for object detection as it has the advantage of being scalable in number of objects and an efficient GPU implementation is possible.

Chapter 3

Temporal framework for RGBD cloud Segmentation

Our segmentation framework is built upon dense mapping methods like Kinect Fusion proposed by [32] and Kintinous by [44]. These methods compute dense depth maps, and pose information for every frame using Iterative Closest Point or FOVIS (Fast Odometry from VISion) by [21] in case of Kintinous. We use depth information to generate vertex and normal maps and segment a given scene using segmentation method by [12]. We use a modification of the edge weight metric proposed by [23]. Using the camera pose information we fuse segmentation maps from different viewpoints in a 3d cloud in an online fashion. The fusion of segments is treated as a region merging problem and is solved by deleting edges from a weighted graph built from segmentation maps to be merged. The segmentation maps are used to discover new objects and to identify objects in real time in a new scene. The segmentation and detection pipeline for the proposed method is shown in 3.1. In the following sections we will discuss each module in detail.

3.1 Data acquisition

As discussed in previous chapters sensors like Microsoft Kinect have led to the development of numerous techniques for RGB-D object detection, mainly because of low cost and ease of use. We use the Kinect sensor for acquiring RGBD data although our method is not just limited to Kinect. But as dis-

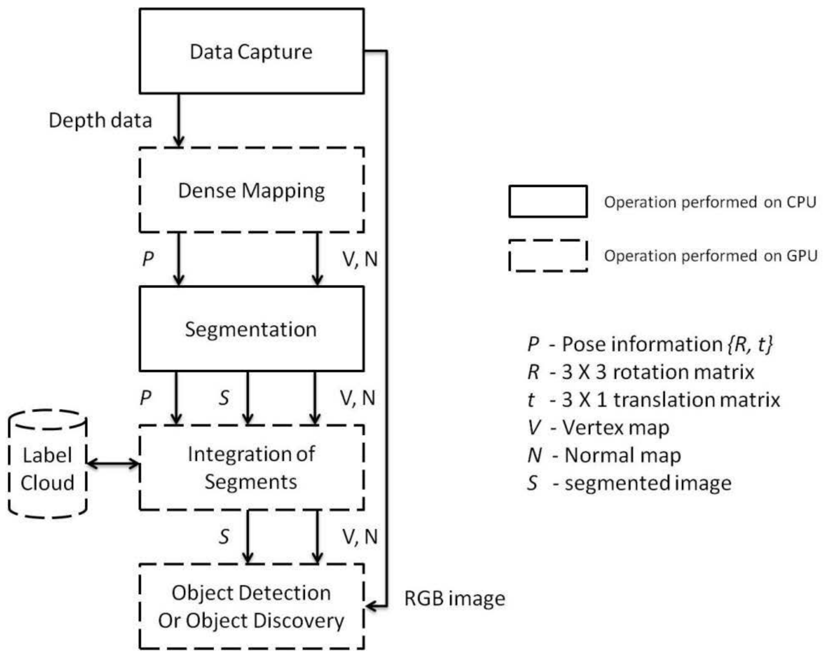

Data Capture
Depth data
Dense Mapping
P
V, N
Segmentation
P
S
V, N
Label Cloud
Integration of Segments
S
V, N
Object Detection Or Object Discovery
RGB image
Operation performed on CPU
Operation performed on GPU
P - Pose information {R, t}
R - 3 X 3 rotation matrix
t - 3 X 1 translation matrix
V - Vertex map
N - Normal map
S - segmented image

Figure 3.1: Segmentation and detection pipeline

cussed in the previous sections the sensor also has some limitations like noisy depth data, missing information due to shadows and infrared absorbing surfaces. To overcome these limitations in literature various methods have been used. Dense mapping algorithms comes under this category. Dense mapping algorithms aggregate depth information into a TSDF (Truncated Signed Distance Function) cloud to minimize noise and fill in depth information as the sensor is moving around.

We use dense mapping algorithms to obtain depth maps. The advantage of using dense mapping algorithm instead of alternative approaches is that besides removing the noise in the data it also provides with pose estimates. Also since iterative median filters fill in information based on neighbors, in case of complex objects or in case there is large amount of information missing, it will lead to incorrect depth maps. Whereas dense reconstruction methods fill in depth information in an incremental fashion from real world data which has the advantage of being more accurate.

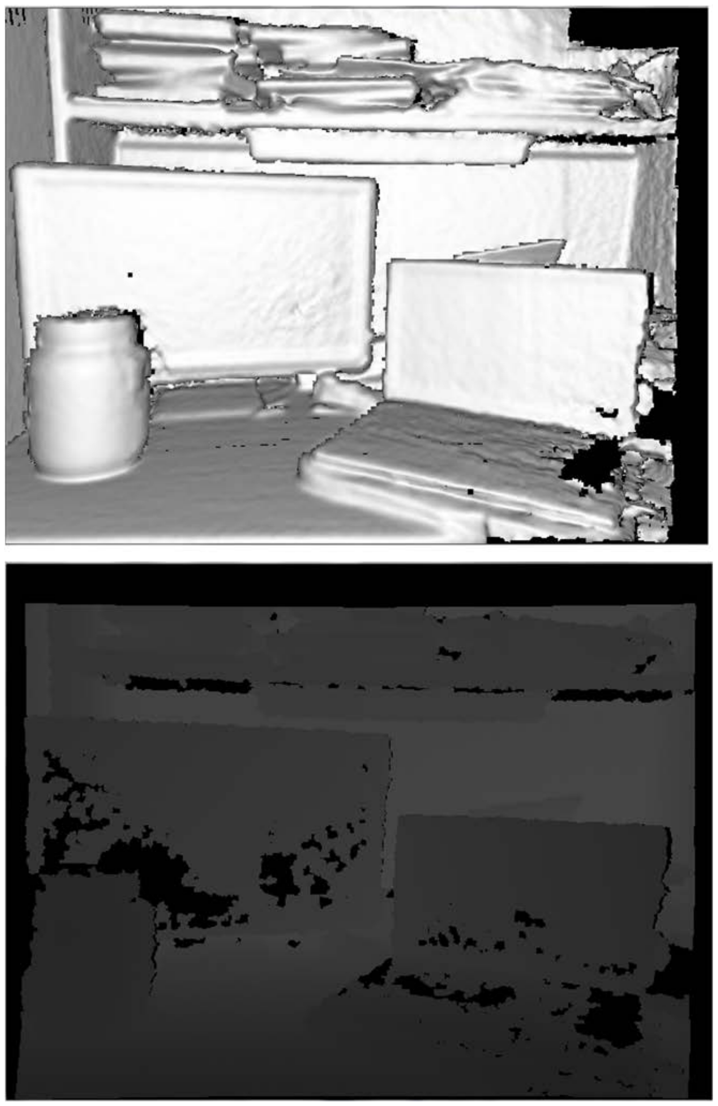


Figure 3.2: Example of dense reconstruction of a given scene. Top images is rendered dense depth map which is the output from Kinect fusion. Bottom image is raw depth image at a time instance t. Note the missing depth information in the raw depth image on the and the information has been filled in reconstructed image.

3.2 Segmentation

Segmentation and multi-viewpoint integration is the central idea of this work. The primary requirement for our work is that segmentation should operate in real time, work in an incremental fashion, should be able to handle missing data and noise and should be able to integrate segments from multiple viewpoints. Another requirement that we impose is that segmentation method should have minimum number of input parameters in order to limit the complexity of the over all system. In this work we use segmentation method by [12] because of its simplicity, computational efficiency and accuracy. Also there is just one parameter for controlling segmentation thus making it suitable for our task.

3.2.1 Efficient graph based segmentation

In this section we will discuss the segmentation problem for a single image as defined by [12]. Given an RGB image I and corresponding depth image D a graph G is constructed as follows. For each pixel a vertex is created and edges are created between neighbors of a pixel. Each edge between vertices v_i and v_j weight w_{ij} is computed which is a non-negative measure of dissimilarity between the two pixels.

The predicate for dividing the graph into various components $C \subseteq V$ where V is set of all vertices or pixels, thus achieving segmentation measures the inter-component dissimilarity to intra-component dissimilarity. The intra-component dissimilarity for component C is defined as follows

$$Int(C) = \max_{e \in MST(C,E)} w(e) \tag{3.1}$$

where MST is minimum spanning tree of component C and above function selects maximum weight of edge e in the MST. The inter-component dissimilarity is defined as minimum weight of the edges between two components C_1, C_2 where $C_1, C_2 \in V$.

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w((v_i, v_j)) \tag{3.2}$$

The pairwise comparison predicate between two components is defined as follows

$$D(C_1, C_2) = \begin{cases} \text{true} & \text{if } Dif\,(C_1, C_2) > MInt\,(C_1, C_2) \\ \text{false} & \text{else} \end{cases} \tag{3.3}$$

Where minimum internal difference $MInt$ is defined as

$$MInt(C_1, C_2) = min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2)) \tag{3.4}$$

Where $\tau = k/|C|$ requires stronger evidence for a boundary for small components. Constant k controls the size of segments and larger k causes larger components. k is the only input to this segmentation algorithm and controls between over and under segmentation. The algorithm is easy to implement and give good run time performance even on a CPU. It is implemented using union by rank and path compression algorithm.

3.2.2 Edge Weights

In the previous section we discussed the segmentation algorithm. The segmentation problem is posed as dividing a graph into components. Each edge in the graph is assigned a dissimilarity weight measure computed neighboring vertices or pixels.

[23], [13] used convexity based criteria as edge weights for graph based segmentation algorithm [12]. We use similar approach for segmentation but with slight modification to the edge weight metric which is defined as follows

$$w_{ij} = \begin{cases} \text{MAXWGHT} & \text{if } d\,(v_i, v_j) \geq d_{min} \\ (1 - n_i \cdot n_j)^2 & \text{if } d\,(v_i, v_j) < d_{min} \text{ and } (v_j - v_i) \cdot n_j > 0 \\ 1 - n_i . n_j & \text{else} \end{cases} \tag{3.5}$$

Where MAXWGHT is a large constant weight, v_i is a vertex point, n_i is the corresponding normal and $d\,(v_i, v_j)$ is the euclidean distance between two vertex points. Rational for deviating from the approach in [23] is explained with 3.3.

3.3 Spatio-temporal Integration of Segments

Spatio-temporal segmentation is the central idea to our approach. In literature there have been some methods which deal with the problem in 2d images. [47]

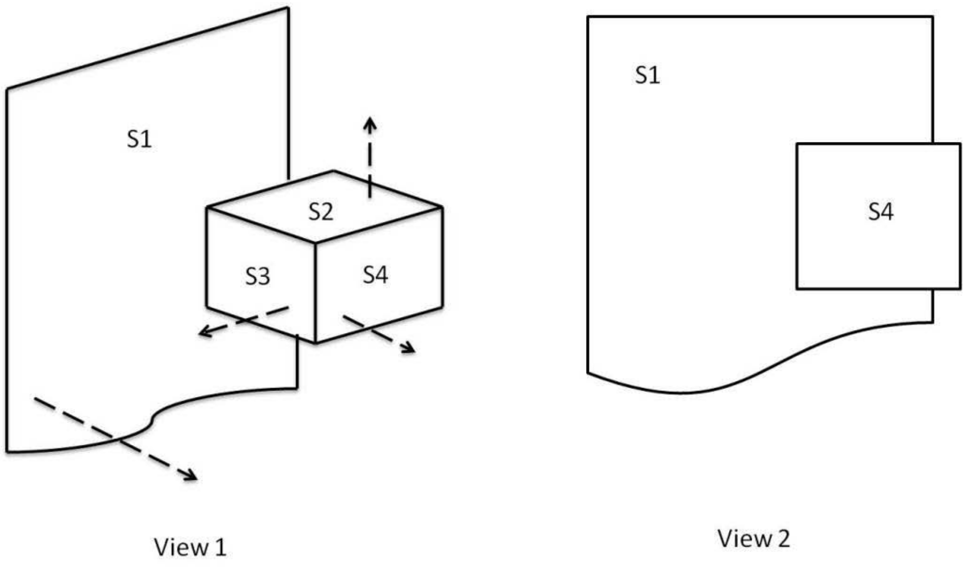


Figure 3.3: View 1 shows two objects, dotted arrows show direction of normals. View 2 shows the same object configuration from a different view. Since surface 4 (S4) has same normal direction as surface 1 (S1) which merges both regions even though they belong to two different objects. Adding a distance based threshold allows to separate two surfaces.

used DAG based approach for detection and segmentation of primary object in video. Motion and intensity are central cues to the approach to perform proposal prediction and to expand the set of object proposals for a frame. [31] used Kolmogorov-Smirnov test to determine spatial similarity and merge regions using weighted directed graph. Such an approach has multiple advantages as compared to segmentation of objects in a single image due to the presence of multiple cues. Motion has been one of the important cues for identifying novel objects in a scene. Also due to the temporal information present multiple frameworks can be used for validating the detections performed.

[23], [13] generate multiple segmentation maps using different thresholds to detect multiple objects. In our work instead of generating multiple segmentations of the same scene we over-segment a given scene and rely on spatio-temporal segmentation to fuse segments of geometrically connected regions. We show that by integrating segments from multiple views allows us to segment complex objects which do not necessarily comply with convexity based criteria in one or more views. Other advantages of an online approach as

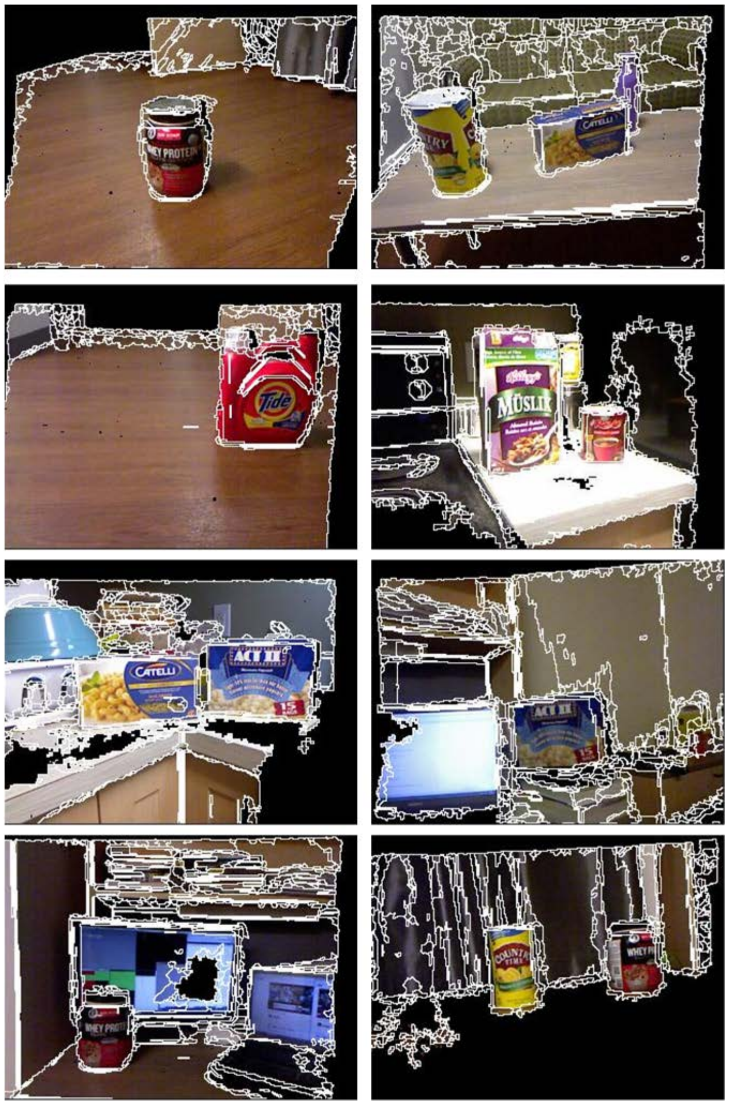


Figure 3.4: Segmented images using the efficient graph based segmentation. The spatio-temporal integration has not been done and segmentation results are for a single frame only. Notice the noise on the edges of the objects due noisy depth data.

compared to an offline approach are as follows.

- Run time performance and scalability - The primary emphasis of this work has been on run time performance of RGB-D cloud segmentation and scalability of the whole architecture. In previous sections we have discussed various methods to perform the task of segmentation. Common approach in contemporary work on RGB-D cloud segmentation have been to capture the complete scene first using a dense reconstruction algorithm like Kinect fusion and then performing segmentation followed by further processing like object discovery. But one limitation of such an approach is that it is not scalable. As the size of the area to be processed increases, the size of the corresponding cloud also increases. One way to decrease the computation time is to downsample the cloud, however one disadvantage of downsampling is loss of resolution and bad performance of segmentation around thin structures.

  We over come the limitation of run time performance and scalability by proposing an online approach for segmentation. Our proposed solution is independent of the amount of surface being processed thus achieving scalability. For segmentation rather than segmenting a dense point cloud collected offline at once, we segment each RGB-D frame independently followed by integration into a segmentation cloud. We also heavily parallelize various components of the segmentation process on a GPU thus achieving real time performance for segmentation framework. Run time analysis for our algorithm is as follows.

  The segmentation method that we use, proposed by [12] has a runtime complexity of $O\left(nlogn\right)$. The algorithm depends on the number of vertices in the graph or number of vertices in a point cloud. In our case n is a constant. Assume w and h be the width and height of the RGB-D images being processed. We use $w = 320$ and $h = 240$ in our work. Thus segmentation cost in our case is $O\left(n^{*}logn^{*}\right)$ where $n = w * h$. The total cost of segmentation framework is $2 * projection\_cost + O\left(n^{*}logn^{*}\right) + merging\_cost$ where $projection\_cost$ is cost of projecting the previous segmentation map from current point of view and storing the integrated cloud which is again a constant time operation and is performed on a GPU. *merging_cost* is cost of merging two segment maps as discussed in the following sections. Again *merging_cost* depends on number of segments in an image which almost remains constant and depends upon segment threshold and complexity of a given scene. Segmentation cost remains constant irrespective of the size of the physical area being processed for a given scene.

- Higher resolution of data - We gain this advantage because of scalability and real-time property of the proposed method. For the task of object detection more information and finer details can be extracted from a high resolution images. For example in texture analysis or while using features like HOG, high resolution will help to extract more discriminative features. By using offline methods where a dense point cloud is collected before processing can only deal with limited resolution to lower the computational cost. Thus the number of voxels per meter are lower whereas by using the proposed method this limitation can be overcome as the method is not dependent on the size of the cloud but only depends on the images being processed.

  Another limitation of offline approach is low resolution of RGB data and due to averaging leads to blurring of resulting images. As discussed in previous sections dense reconstruction methods collect RGB-D data and integrate the information into a dense cloud by running weighted average process. Due to this averaging process and minute errors in the integration process results in RGB data to be heavily smoothed and also loss of edge information. This can result in bad performance of color based segmentation or image description methods. Also to make offline methods computationally efficient low resolution data is used, which also results in low resolution of the RGB data.

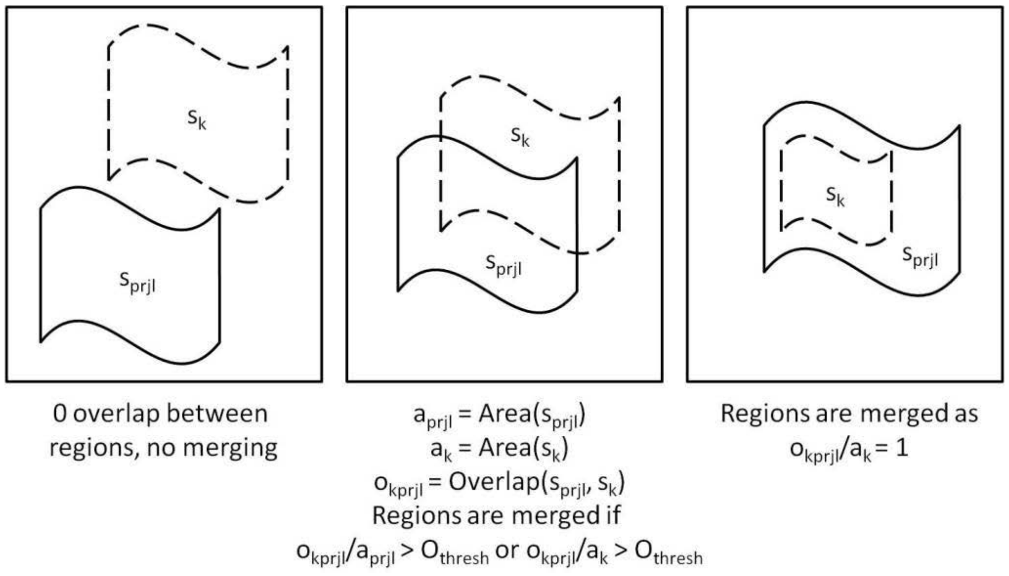
- Applications in robotics - Dividing an image into meaningful segments has numerous applications. There have been considerable interest in the research community to achieve segmentation which is closer to the true object boundaries. With advances in robotics and depth sensors there is considerable interest in using this additional information for tasks related to grasping, object manipulation and navigation. Segmentation plays an important role in tasks for example for the task of grasping it is important that the structure of the object is precise and the object of interest can be separated from the background. But due to noisy depth data and missing information it is not suitable to use depth data directly. The proposed segmentation framework provides dense depth information and meaningful segment(s) for objects. The framework is also flexible to support an unsupervised or a supervised approach as segmentation can be easily coupled with user interaction.

- Amodal completion - We also deal with the problem of amodal completion. While performing segmentation in an indoor environment due to heavy occlusion it is possible that one single object gets divided into multiple regions. [16] proposed a solution to this problem by geometric fitting. We we demonstrate that our framework can perform amodal completion implicitly by integrating information from multiple views.

Our spatio-temporal segmentation algorithm is described as follows. Let p_t be the pose information given at time t, and V_t and N_t the vertex map and normal map at time t obtained from dense mapping. Using pose information p_t we project label information from dense label cloud which has the integrated label information from time 0 to $t-1$.

3.3.1 Merging Criteria

Before merging multiple segments from different segmentation maps, we determine the extent of overlap between the segments. Merging criterion between



s_k
s_{prjl}
0 overlap between regions, no merging
a_{prjl} = Area(s_{prjl})
a_k = Area(s_k)
o_{kprjl} = Overlap(s_{prjl}, s_k)
Regions are merged if
$o_{kprjl}/a_{prjl} > O_{thresh}$ or $o_{kprjl}/a_k > O_{thresh}$
Regions are merged as
$o_{kprjl}/a_k = 1$

Figure 3.5: Assignment of edge weights. Given two segments s_k from S_t and s_{prjl} from S_{prj_t}.

two segments s_i and s_j is defined as follows

$$
\begin{aligned}
o_{ij} &= \text{Overlap}\,(s_i, s_j) \\
a_i &= \text{Area}\,(s_i) \\
a_j &= \text{Area}\,(s_j) \\
\text{MergeRegion}\,(s_i, s_j) &= \begin{cases} 1 & \text{if } o_{ij}/a_i > O_{thresh} \text{ or } o_{ij}/a_j > O_{thresh} \\ 0 & \text{else} \end{cases}
\end{aligned}
$$

Given a segment s_i, *Area* computes the number of pixels having label i in a 2D image. *Overlap* computes the overlap between two segments s_i, s_j by computing number of 2D coordinates in segmentation images S_t having label i and S_{prj_t} having label j.

3.3.2 Merging Segmentation Maps

Given two segmentation maps the aim of spatio-temporal integration is to merge the segments in a consistent manner such that object boundaries are preserved. As mentioned previously that we oversegment the image. Which leads to division of a single object into multiple segments. Assuming that these oversegmented regions preserve the object boundaries, goal of this step is to preserve the object boundaries and integrate segmentation maps from

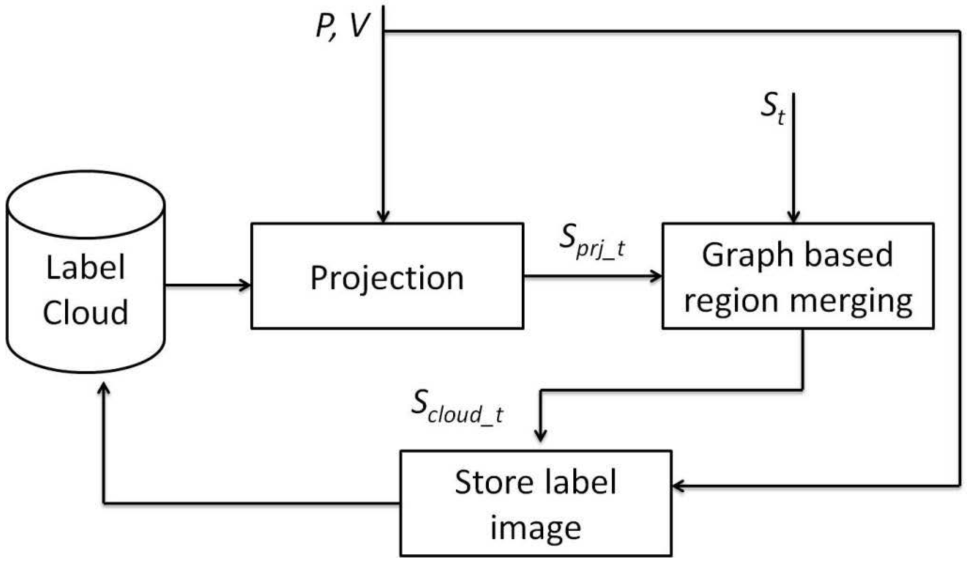


Figure 3.6: Merging of segmentation maps as the Kinect camera moves around. P is the pose estimate and V is the vertex map.

different viewpoints into one consistent label cloud.

Integration of segmentation maps is formulated as region merging problem. A graph is created from the segmentation maps to be merged and edges are deleted from the graph. Regions which stay connected after edge deletion are labeled as one single region. Let S_t be the segmentation map at time t and S_{prj_t} be the projected segmentation map obtained from label cloud from the current pose. Both segmentation maps are used to generate S_{cloud_t} which is stored back in the label cloud. The merging process is explained pictorially in 3.6.

Let m be the number of unique labels in segmentation map S_t and $s_0, s_1, \cdots, s_m$ be the corresponding segments. Similarly n is the number of unique labels in segmentation maps S_{prj_t} and $s_{prj0}, s_{prj1}, \cdots, s_{prjn}$ are the segments. The graph is constructed in the following manner. For each segment in both segmentation maps a graph node is created. Edges between nodes from segmentation map S_t and nodes from S_{prj_t} are created and each edge is assigned a weight computed from overlap metric o_{ij}. Note that the m segments from S_t don't have any edge between them as there is no overlap between any segment. The same is true for n segments of S_{prj_t}.

To perform merger of the segments the resulting graph is traversed. Cycles are avoided by keeping track of the nodes traversed. Any pair of segments having overlap greater than the overlap threshold are merged in the graph and are assigned same label. For the connected nodes in the graph its label is determined by selecting label from the node corresponding to the segment having largest area. Rational for selecting the segment having largest area is explained later in this document. Now some of the nodes will still have unassigned label and for these nodes a new label is generated, creating a new label.

3.3.3 GPU Implementation

In the previous sections we have discussed segmentation of RGBD cloud and multiview integration of RGBD data to obtain dense segmentation map for a scene. We operate on 512 X 512 X 512 label cloud which maps to 3m X 3m X 3m real world space. The projection of the label cloud from current viewpoint and update of the cloud is performed for every frame. Updating every voxel in the label cloud or getting the projection of the cloud is computationally very expensive. In order to achieve real time performance these tasks are implemented on the GPU. The projection of the cloud from the current viewpoint is performed as follows. Let R, t be the rotation and translation matrix information obtained from pose estimation. The size of each voxel is csz, csz, csz where $csz = 3.0/512$. Given the vertex map V corresponding to current viewpoint and each vertex in the vertex map be v_i. For each vertex v_i the voxel index is determined as follows

$$voxel_i.x = float2integer(v_i.x/csz)$$
$$voxel_i.y = float2integer(v_i.y/csz)$$
$$voxel_i.z = float2integer(v_i.z/csz)$$

To parallelize the operation for each vertex a thread on a GPU is launched and the voxel index is computed which is used to project label cloud to a $2D$

image which represents segmentation from current point of view. Similarly, after performing region merging operation a new consolidated segmentation map is obtained. This consolidated segmentation map at time t S_{cloud_t} is added to the dense label cloud *LabelCloud*. The size of the *LabelCloud* is $VOLUMEX, VOLUMEY, VOLUMEZ$. This operation is also performed on the GPU. Let R, t be the rotation and translation matrix information obtained from pose estimation and R_{inv} be the inverse of R. Let f_x, f_y be the focal length and c_x, c_y be the image center. On a GPU for each possible value of $\nu.x$ and $\nu.y$ is launched where $1 \leq \nu.x \leq VOLUMEX$ and $1 \leq \nu.y \leq VOLUMEY$. Each thread runs in parallel and Algorithm 1 is performed for each thread.

Algorithm 1 Label Cloud Update

```
 1: procedure UPDATE($\nu$, $V$, $S_{cloud\_t}$, $f_x$, $f_y$, $c_x$, $c_y$, $csz$, $R_{inv}$, $t$, $LabelCloud$,
    $cols$, $rows$, $distthresh$)
 2:     $\nu.z = 1.$
 3:     loop:
 4:     if $\nu.z > VOLUMEZ$ then goto exit
 5:     $v_g \leftarrow GetGlobalCoordinates(\nu, csz).$
 6:     $v \leftarrow R_{inv} * (v_g - t)$
 7:     if $v.z \leq 0$ then
 8:         $\nu.z \leftarrow \nu.z + 1$
 9:         goto loop
10:     $coo.x \leftarrow float2integer(v.x * f_x / v.z + c_x)$
11:     $coo.y \leftarrow float2integer(v.y * f_y / v.z + c_y)$
12:     if $coo.x \geq 0$ and $coo.y \geq 0$ and $coo.x < cols$ and $coo.y < rows$ then
13:         $p.x \leftarrow V(coo.x, coo.y).x$
14:         $p.y \leftarrow V(coo.x, coo.y).y$
15:         $p.z \leftarrow V(coo.x, coo.y).z$
16:         $dist \leftarrow norm(p - v_g)$
17:         if $dist < distthresh$ then
18:             $LabelCloud(\nu.x, \nu.y, \nu.z) \leftarrow S_{cloud\_t}(coo.x, coo.y)$
19:     $\nu.z \leftarrow \nu.z + 1$
20:     goto loop
21:     exit:
```

Algorithm 2 Compute global coordinates

```
1: procedure GetGlobalCoordinates($\nu$, $csz$)
2:     $p.x \leftarrow (\nu.x + 0.5) * csz$
3:     $p.y \leftarrow (\nu.y + 0.5) * csz$
4:     $p.z \leftarrow (\nu.z + 0.5) * csz$
5:     return p
```

Chapter 4

Experiments

In this section we discuss the main results of this work. The experiments have been performed on Intel Xeon 6 core processor with NVIDIA Tesla C2075 GPU card. Microsoft Kinect was used for data acquisition. In this chapter we will evaluate the performance and properties of the proposed segmentation framework.

4.1 Segmentation performance

We evaluate the proposed segmentation framework on NYU2 RGBD dataset by [36]. The dataset consists of 1449 RGBD images, gathered from a wide range of commercial and residential buildings in three different US cities, comprising 464 different indoor scenes across 26 scene classes. The images were hand selected from 435,103 video frames, to ensure diverse scene content and lack of similarity to other frames. We compare the results for the proposed method with [20]. We use the evaluation criteria used in [20] to quantify the error between generated segmentation map and ground truth. The evaluation critria is as follows. The boundary edges are extracted from the segmentation map, and ground truth segmentation. For each pixel in computed segmentation maps distance to nearest boundary pixels is computed using chamfer distance. Similar distance map is computed for ground truth boundary map. Difference between both the distance maps is computed. The computed difference is summed for all pixels and normalized by product of image width and image height. If w, h is the width and height of image. $d(i)$ is the distance of

Method	Score
[20]	19.35
Proposed method (non temporal)	19.37
Proposed method (temporal)	17.54

Table 4.1: Evaluation of the proposed segmentation framework.

a pixel to a nearest boundary pixel.

$$error = \frac{\sum_i abs\left(d_{gtrth}\left(i\right) - d_{seg}\left(i\right)\right)}{w * h} \tag{4.1}$$

To conduct the experiment we determine videos corresponding to 1449 RGBD images, then we extract 8 frames previous to the each 1449 RGBD image. We run the proposed segmentation framework on the sequence of images. The final segmentation that we achieve for 1449 images is used for evaluation. The segmentation performance is shown in table 1.

The primary reason for our method performing better than [20] is that our segmentation is not affected by color. Segmenting only the depth data gives better performance as segmentation results are not affected by shadow, lighting changes in a scene or reflective surfaces like mirror. The performance of our method can be further improved. Since we run the experiment only on past eight frames for each 1449 frames, the performance can be improved by using more temporal information. Also the performance is little bit lower due to the fact that we still have some missing information in the depth maps due the Kinect camera limitation. Better accuracy can be obtained by operating on closer distance as Kinect loses depth resolution and accuracy as the distance increases. Due to dense depth reconstruction errors there is also a drop in performance. Proposed method will give better results when objects of interests are at a close distance. We also perform qualitative evaluation of temporal segmentation versus non temporal segmentation in figure 4.1 to 4.8.

As mentioned before the quality of the segmentation results can be improved by incorporating more temporal information. The primary reason for this improvement is complex objects can be segmented properly. Also noisy

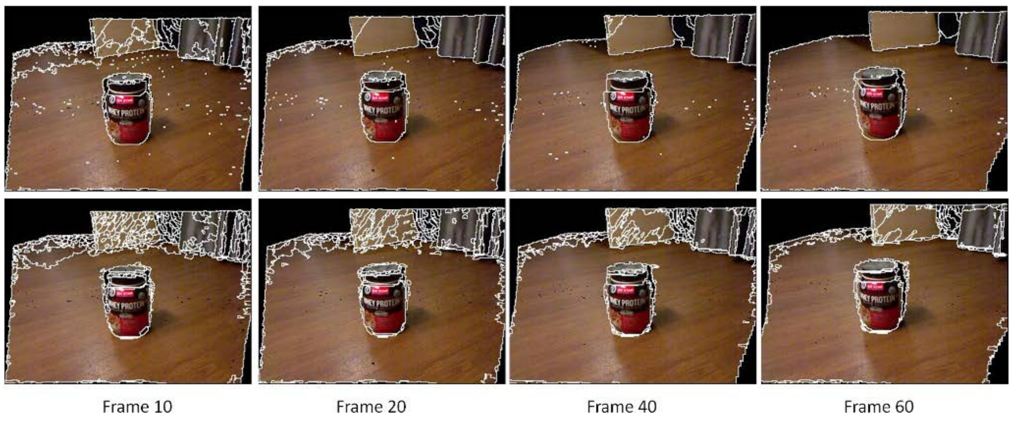


Figure 4.1: Top row shows result of object boundaries using the proposed multiview segmentation method. Bottom row shows segmentation results computed on one image for the same frame. Our method gives more precise and clear object boundaries. Following images show more examples.

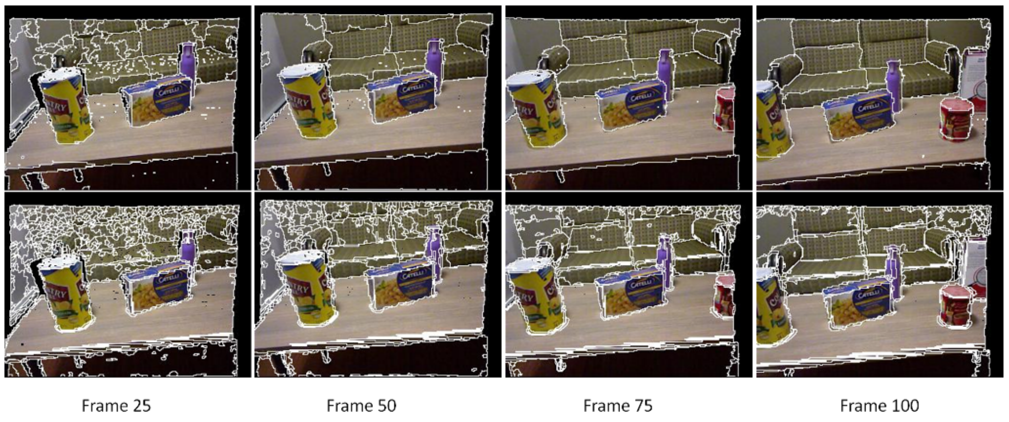


Figure 4.2: Segmentation results

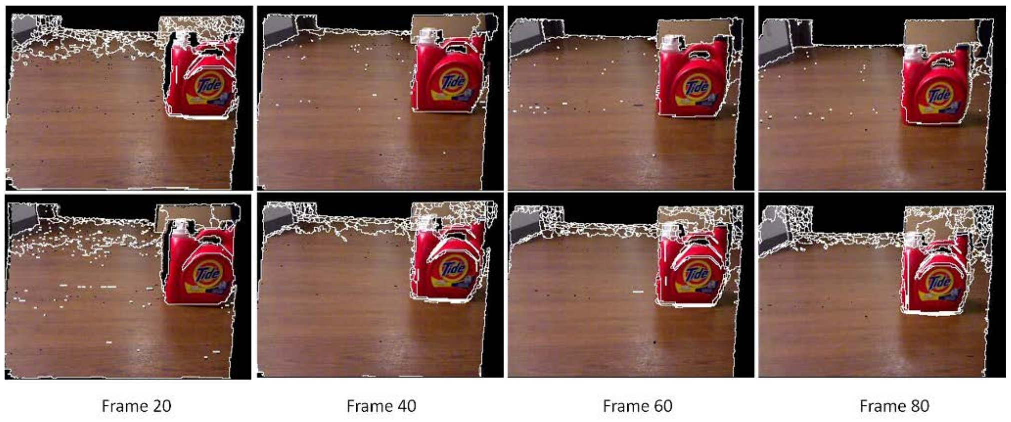


Figure 4.3: Segmentation results

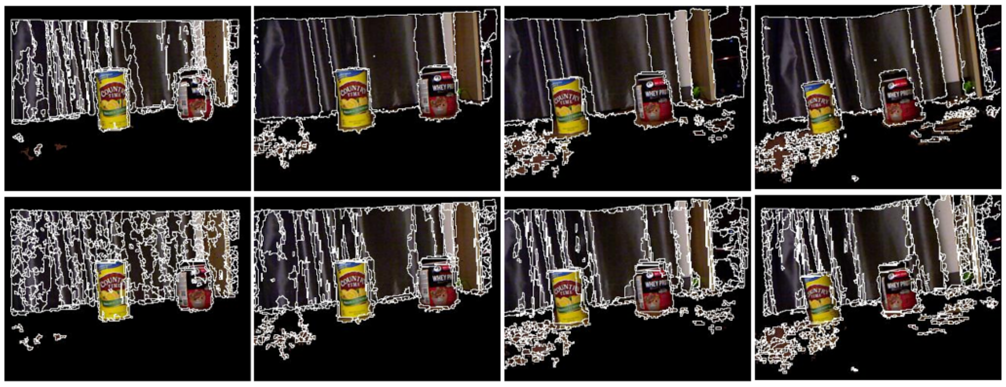


Figure 4.4: Segmentation results

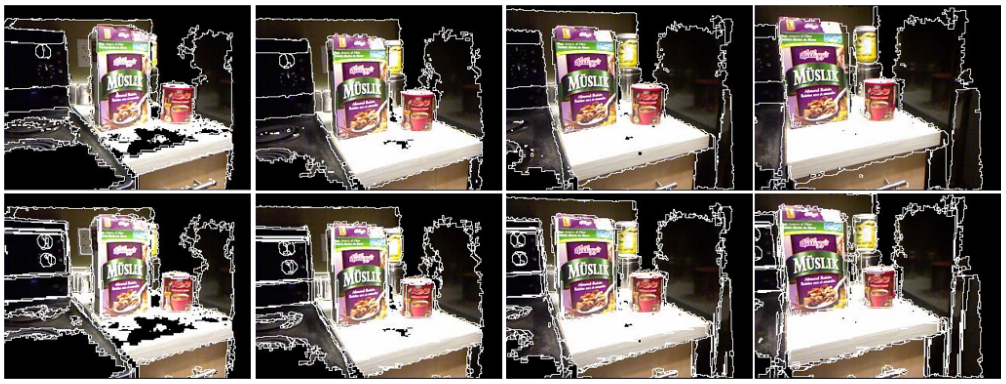


Figure 4.5: Segmentation results

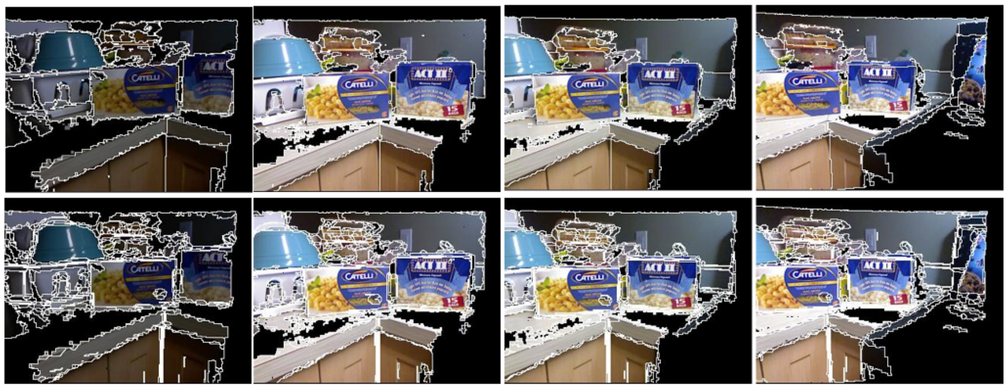


Figure 4.6: Segmentation results

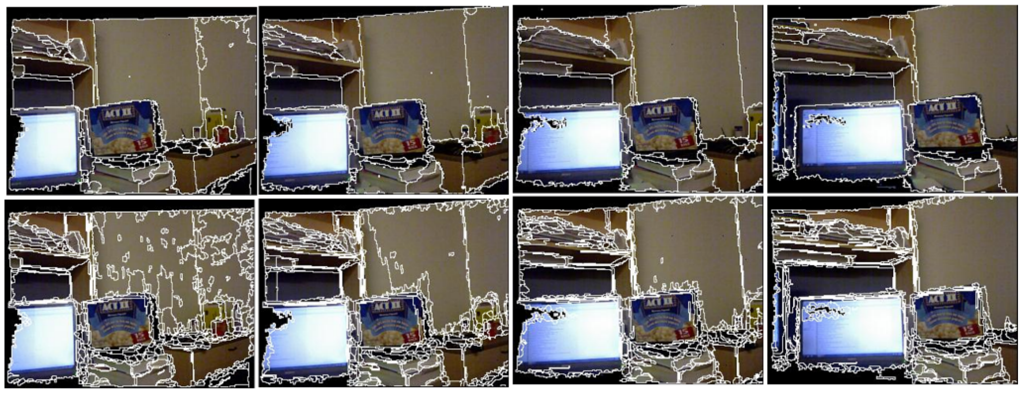


Figure 4.7: Segmentation results

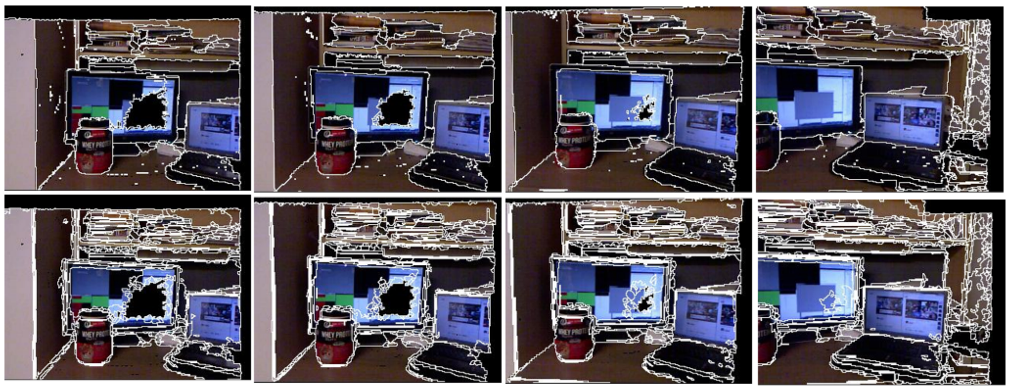


Figure 4.8: Segmentation results

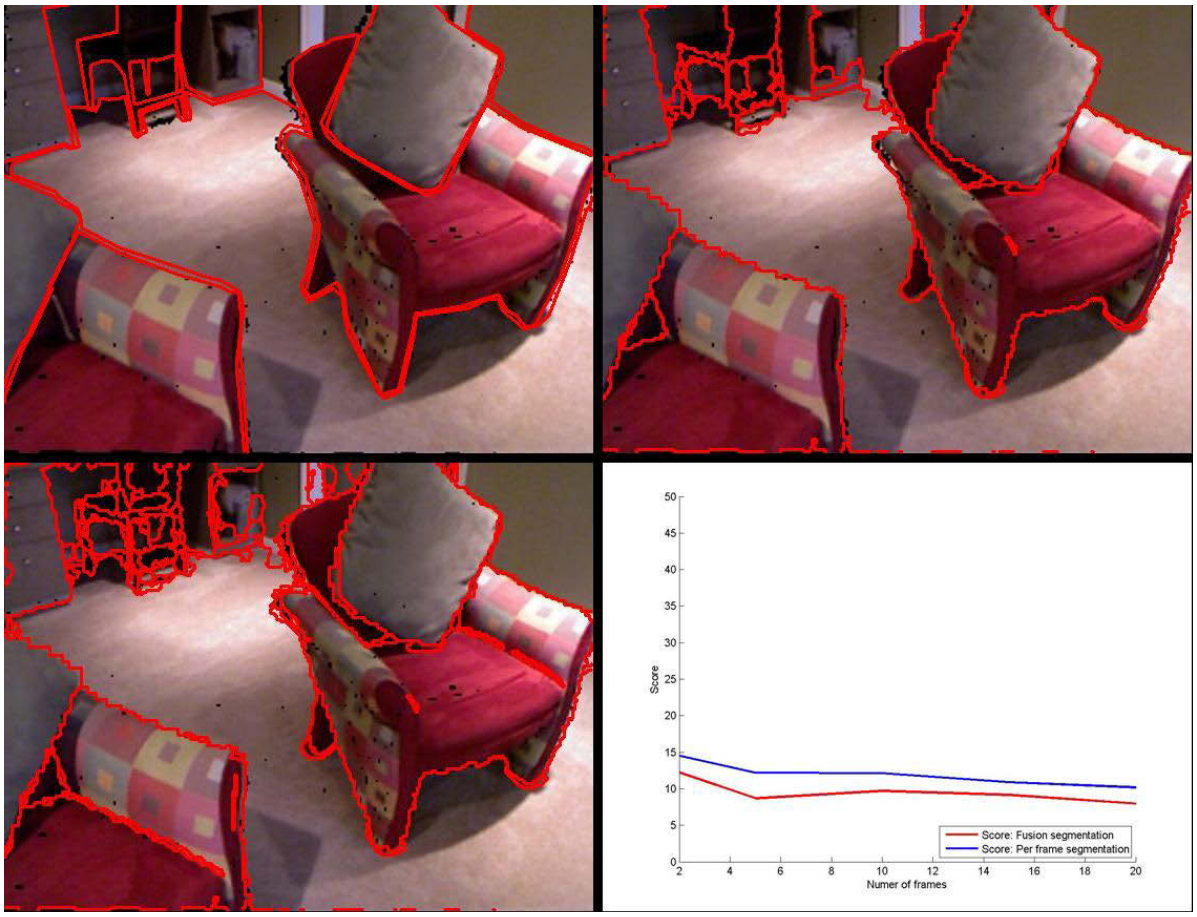


Figure 4.9: Top left image: Ground truth object boundaries. Top right: Segmentation results from proposed temporal segmentation Bottom left: Segmentation of single frame. Bottom right: segmentation performance vs number of frames

and missing information from Kinect which often causes drop in performance. This can be reduced with more temporal data. To verify the claims we conduct experiments to show relation between segmentation performance and number of frames. For this purpose we compare segmentation performance of proposed method with segmentation on a single frame to emphasize that performance improvements are not due to smoothing effect of depth maps by Kinect fusion. For every 1449 RGBD image we extract n frames from the corresponding videos where $2 \leq n \leq 20$. The results are shown in figure 4.9. More results are shown in Appendix 1.

4.2 Frame rate

The main emphasis of this work has been on RGBD cloud segmentation. We leverage GPU processors and proposed multi-view temporal segmentation algorithm to achieve a real time performance. Existing methods, which we refer to as batch or offline methods, first collect a dense map of an entire scene before segmenting the integrated point cloud. It can become difficult to segment large volumes of data as the number of points increase. Where as we adopt an approach of segmenting only the current point cloud and merge its segments with those that have been segmented so far. In other words, our algorithms works at the segment level, instead of the point level, and considerably reduces the complexity of the solution and is not dependent on number of points in the dense depth cloud. We integrate raw depth maps in a dense volume cloud and take projection of the cloud from current point of view to obtain less noisy and dense depth information. This allows us to segment the scene in an online fashion and run time of the segmentation remains the same as the size of projected depth maps are constant. For our implementation we are able to achieve a segmentation rate of at 5-6 fps on images on resolution of 640X480. Recently [20] proposed similar method which operates at 0.8 fps for images of same resolution.

4.3 Segmentation of complex objects

In this section we evaluate the performance of the proposed segmentation method. As we can segment the scene in an online fashion, our segmentation method is able to handle more complex objects which violate the convexity criteria in one or more of the views. Most of the common objects present in indoor scenes are complex and not necessarily convex in one or more views, which might lead to over-segmentation of a given object. But we leverage the capability of integrating multiple views to correctly segment a given object. In Figure 4.10 we show the segmentation of a complex object. It can seen that the red box is non convex in some of the viewpoints. But as the sensor

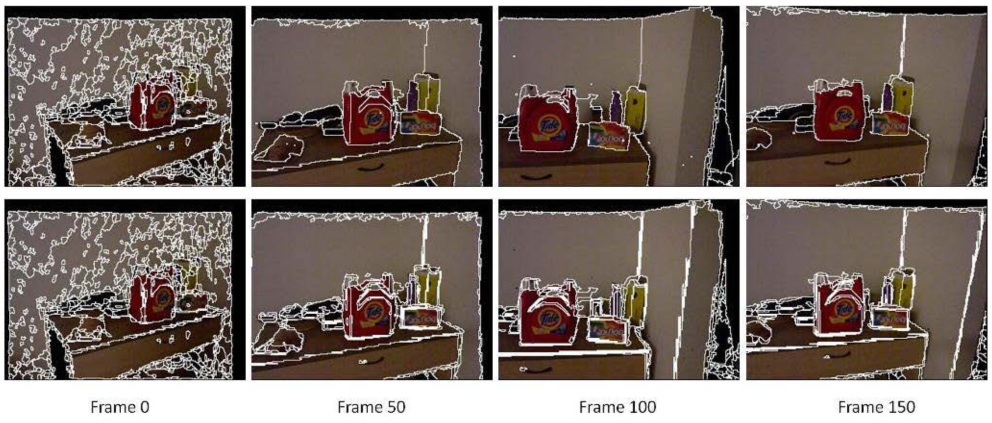


Figure 4.10:

is moving around if a convex connection is discovered between one part of the over segmented object with another the labels are merged leading to correct segmentation of the object.

4.4 Number of constraints imposed

We also compare our method with other methods in terms of the number of constraints imposed for segmenting a scene. [36], [16] uses a gPb contour based approach for segmentation. [36] aligns the captured RGB-D cloud before performing segmentation and each pixel is assigned a plane using CRF. [22] use mutiway cut to perform segmentation which is an NP hard problem and only approximate solution can be obtained. [14] uses a user specified radius and small or large radius can affect the detection of true boundaries of an object. [23] generates multiple segmentations of a given scene. [1] proposed use of fixation points and gPb contours for segmenting an object. They also proposed and extension of original method for segmenting multiple objects by selecting multiple fixation points.

We compare our approach with these methods in terms of complexity of overall approach. Instead of performing complex tasks as plane estimation before segmenting or using complex or approximate segmentation methods, we use a simple segmentation methods and do not impose any alignment constraint on the data. This help us to overcome computational complexity of preprocessing the data and eliminate possibility of errors in plane estimation process. Also instead of selecting multiple thresholds for segmenting the same scene, we over segment every frame and rely on temporal integration to overcome segmentation errors. The rational behind such an approach is, as the segmentation method use convexity criteria, even after over segmenting an RGB-D cloud boundaries are still preserved but due to noise and changing viewpoint overlapping segments are generated in consecutive segmentation maps from different viewpoints. Temporal segmentation merge these overlapping segments thus correctly segmenting a scene and also preserving the object boundaries.

4.5 Amodal completion

Due to occlusion a given object can be divided into multiple segments. The problem of combining these multiple objects into one correct segment is known as amodal completion. [16] handled amodal completion by estimating a parametric geometric model and merging superpiexls into bigger regions based on agreement among parametric geometric models. Instead of computing a geometric model we rely upon information from multiple views. In 4.11 we show some examples of amodal completion by integrating multiple views. As shown in the figure the floor is over segmented as the chair obstructs complete view of the floor. But as the camera moves around connection between various parts of floor are discovered and the floor is combined into one single segment.

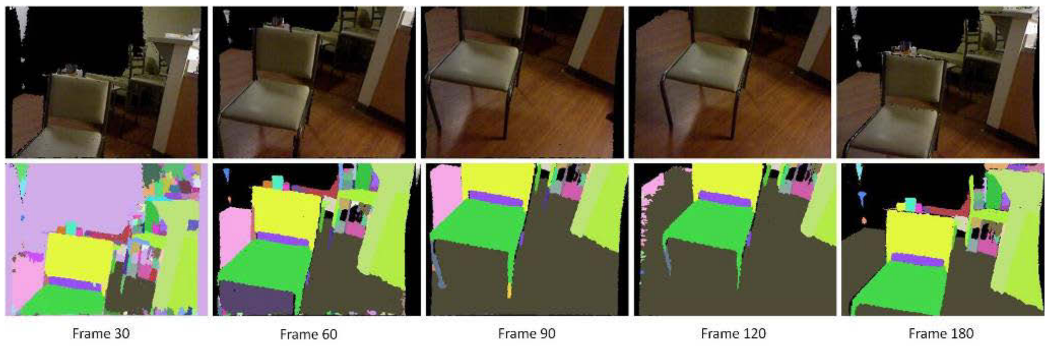


Figure 4.11: Top row shows the RGB image and bottom row shows the corresponding segmentation maps. In the beginning frames we can see that floor is divided into different segments due to occlusion from chair. But once a connection is established between segments they remain connected as seen in Frame 180.

Chapter 5

Applications of the segmentation framework

In this chapter we discuss various possible applications of the proposed segmentation framework. In this thesis we have focused on a robust RGB-D segmentation framework which uses temporal information. We have been emphasizing on real time and online performance the segmentation framework for which we have used GPU. In the following sections we discuss the applications of real time temporal segmentation framework for tasks like object discovery and object detection.

5.1 Object discovery

Object discovery, as the name suggests, deals with discovering objects in a given scene in an unsupervised manner. In literature color, motion, depth or combination of different cues have been used to perform object discovery. Some notable works in this area have been by [33], [13], [42], [19], [18], [23]. There are also active vision based approaches by [29], [22] which rely on selection of some points inside an object.

Our proposed segmentation method can naturally solve the object discovery problem with minimum additional work. Rather than selecting the segments and classifying them as object, we select segments as possible objects which are in the center of view and have been persistent in segment size for a few frames. After segmenting current scene and during the merging of seg-

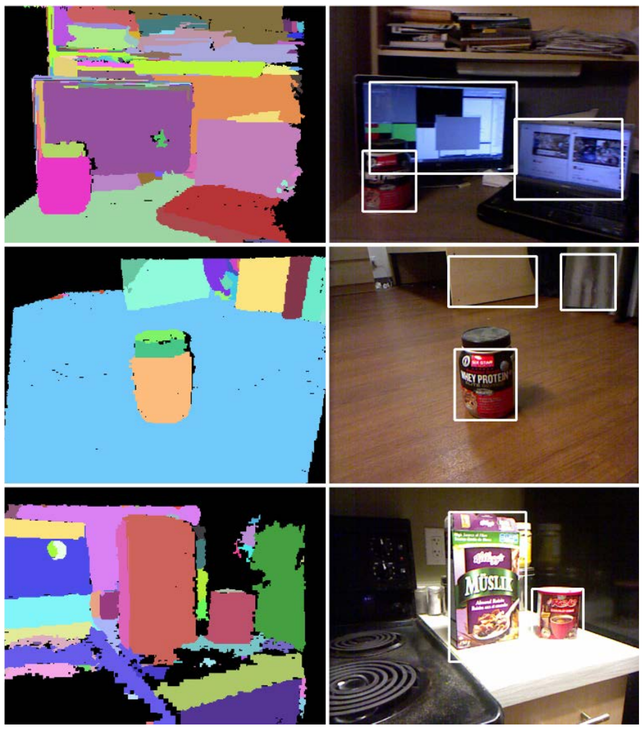


Figure 5.1: Discovered objects in some test scenes bounded by white rectangles.

ments from multiple frames using graph based approach, for all the segments to be merged we assign them the label of the segment having largest area. By assigning the label having the largest area helps in tracking the persistence of size of segments. We also impose size and eccentricity constraint on segments to prune the candidates. Segments which belong to floor or table or other surfaces usually reach edges of the view allowing objects to be delineated in the scene. Figure 5.1, 5.2 shows some of the results for object discovery.

5.1.1 Unsupervised database creation

Besides using the simple approach to object discovery discussed previously our framework also allows use of more advanced object discovery methods. The aim of object discovery in this work is to demonstrate unsupervised generation of an object database. As we have discussed previously that generation of a training database is a difficult procedure often requiring complex setup, also generating ground truth database is a costly process. We demonstrate

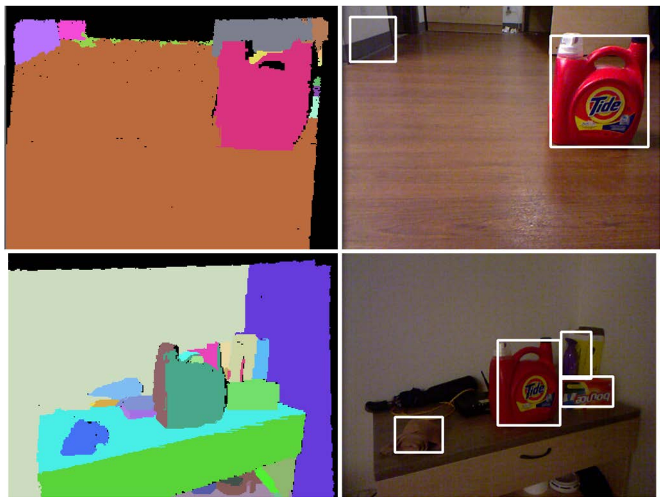


Figure 5.2: Discovered objects in some test scenes bounded by white rectangles.

that the process can be simplified by combining object discovery with online segmentation.

Our segmentation method allows us to segment a given scene by just moving the Kinect camera around. By combining object discovery with dense temporal scanning of a scene we can keep track of the labels and capture an object from different poses. This simplifies the database generation procedure to such an extent that only thing one has to do is pick up a Kinect camera and move it around in an object in any fashion. It also allows to capture data for multiple objects simultaneously. We demonstrate our results for data collection in 5.3, 5.4. [17] developed methods to perform 3D model creation of objects which can also be used for model creation. Figure 5.3, 5.4 shows result for our approach for object database creation.

5.2 Object detection

In this work we have emphasized the need for methods which are scalable and operate in real time to have practical applications. [27] used detection based framework for labeling 3d scenes with good accuracy. The only shortcoming of such an approach is it needs sliding window detectors for every object, and this results in a large search space for every object.

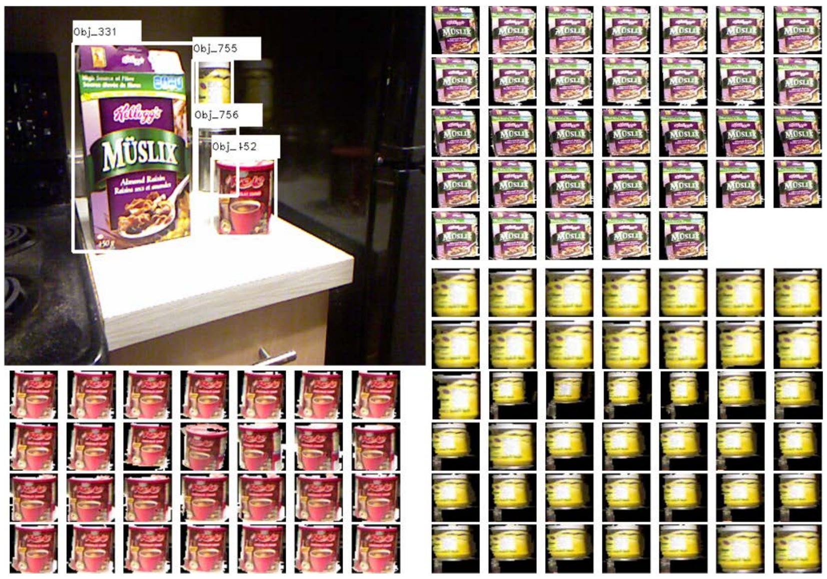


Figure 5.3: Top row shows the objects detected in the RGB image shown at the bottom. Middle row shows some images that are added to the object database as the camera moves around for one object. For all the objects detected similar images are generated and object labels are inferred from segmentation map shown in the bottom row.

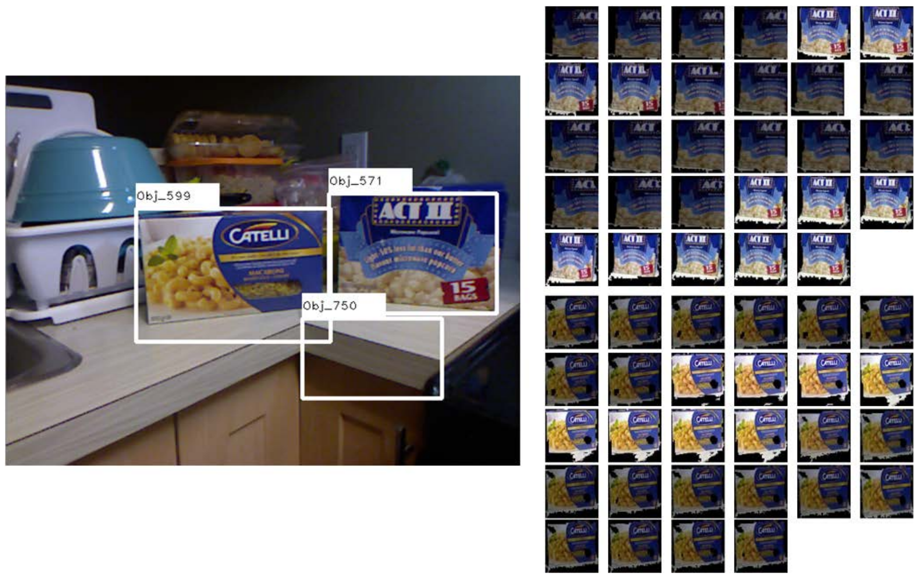


Figure 5.4:

We discuss how a scalable and real time framework can be developed for object detection and 3d scene labeling. We build object detection framework similar to [41]. As discussed, by combing object discovery and proposed segmentation framework we can collect images of objects in an unsupervised fashion. We build our object detector on this framework. We collect images corresponding to objects and build a large database of objects. We convert samples in database from RGB color space to LAB color space and normalize color channels to have unit variance and zero mean. We perform comparison of query images with the images in database using modified SSD metric proposed in [41]. We use GPU implementation of the method to achieve real time performances on a large amount of data. It also allows us to add objects to the database in an online fashion making the detection framework online. Any object that is not detected is added to the object database dynamically.

One limitation of the above nearest neighbor framework is, it is still sensitive to lighting and viewpoint changes. However, the aim of this experiment was to develop a proof of concept and it is possible to construct better object models for detection or even bootstrap the databse with known objects to perform detection. In our experiments we can perform object detection at nearly 15-20 fps with little or no affect on run time as size of database increases.

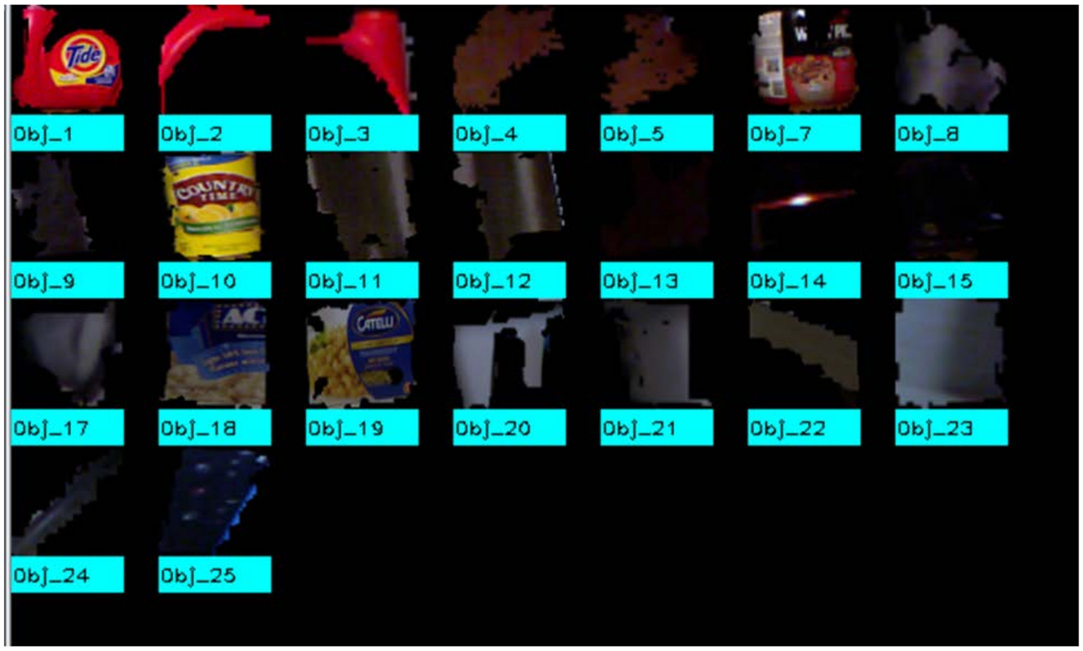


Figure 5.5: Example of object detection using the segmentation framework. The objects collected using object discovery are shown in this image. *Obj_X* is the name of the object generated automatically.

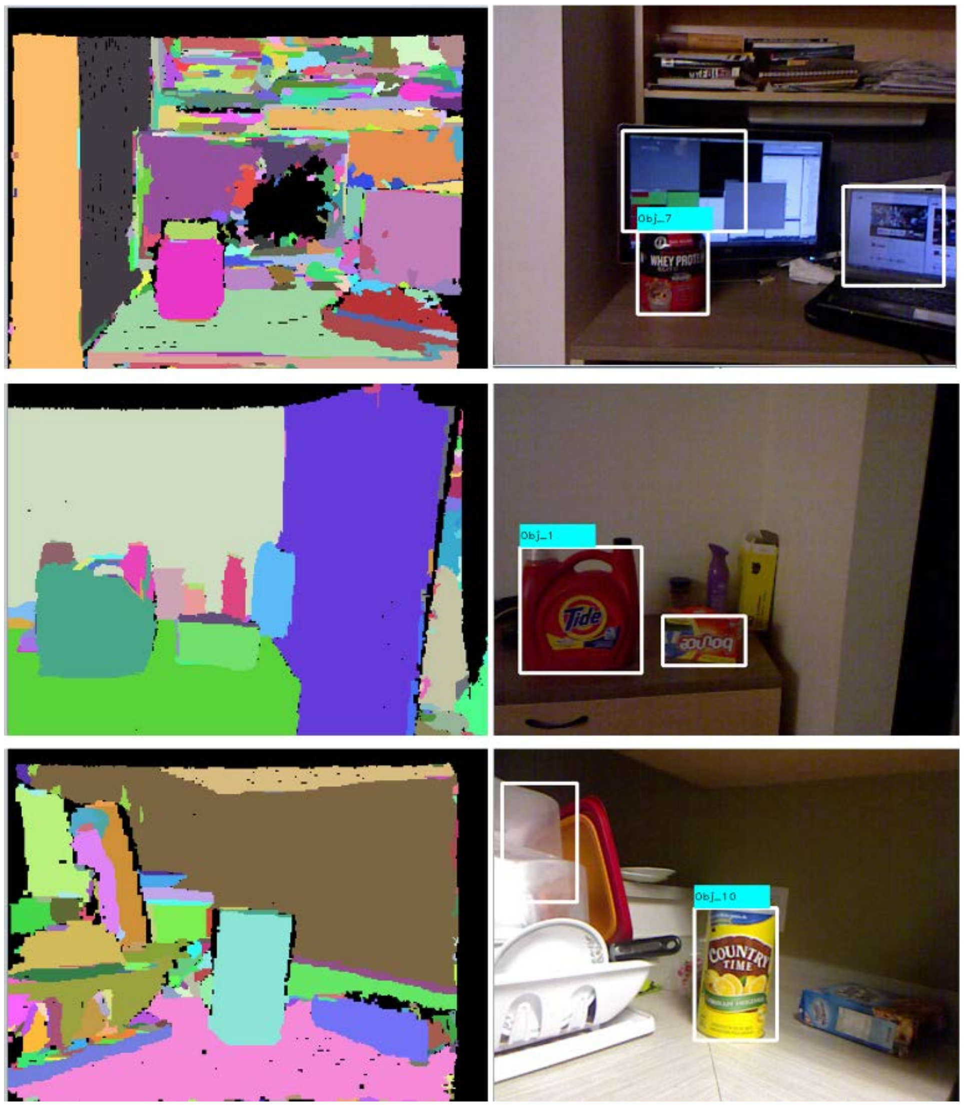


Figure 5.6: Example of object detection. All the object candidates are enclosed with white bounding boxes. Correctly identified objects have cyan label on the top left of the rectangle. Cyan label shows the object name generate by object discovery as shown in 5.5

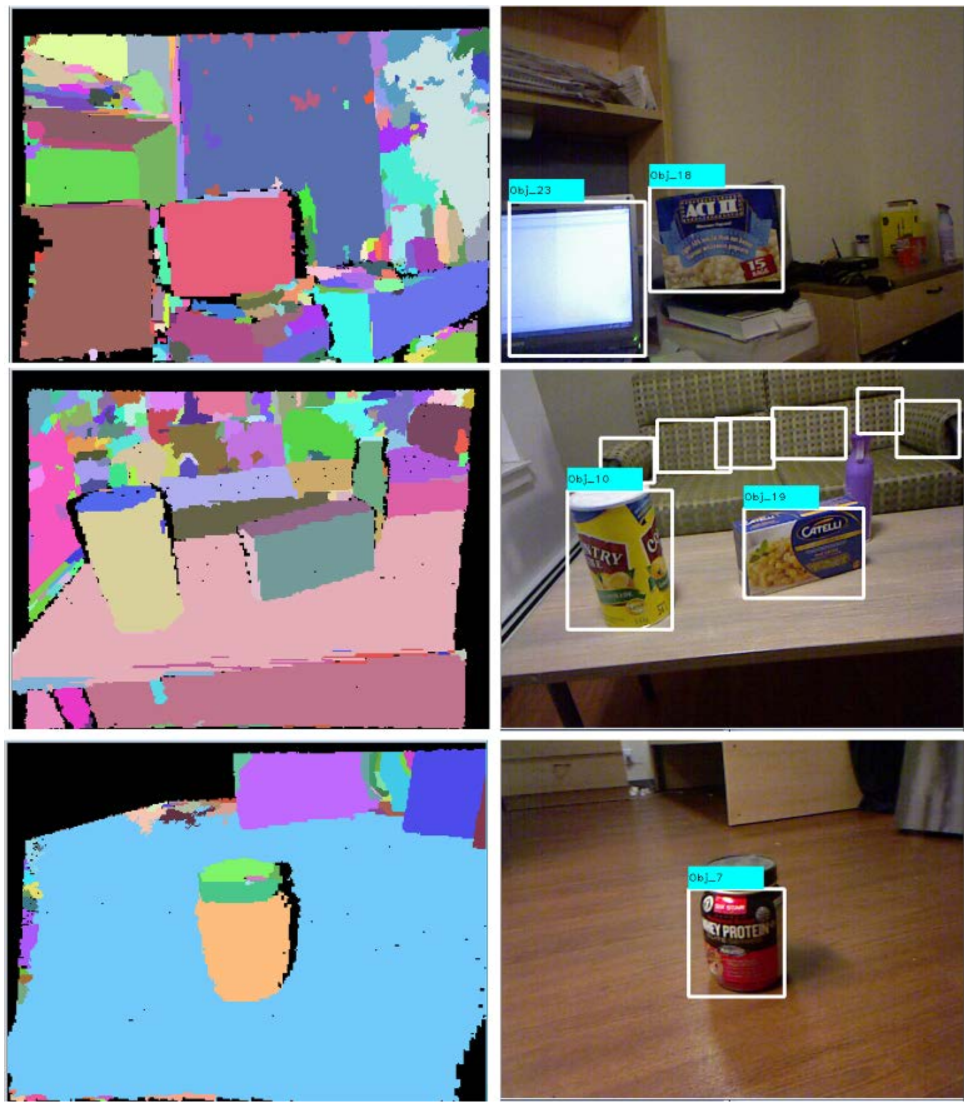


Figure 5.7: Example of object detection.

Chapter 6

Conclusion

The primary focus of this work has been on RGB-D cloud segmentation. We have discussed complexities of the human visual system. It uses various cues in combination to group elements of visual stimuli. Some of the cues used by human visual stimuli are motion, depth, shadow and shading. These cues help us to delineate object boundaries. The existing methods on RGBD cloud segmentation either rely on training database which can be costly to acquire or are slow in operation. Existing methods also being offline do not operate on large amount of surface due to the large number of points in the RGBD cloud. In this thesis we have proposed a temporal segmentation method which operates on 2D RGB-D data and perform multi-view point integration of segment maps to generate a global scene segmentation. We have used an efficient graph based segmentation algorithm as it is fast in operation and does not require many input parameters. We rely on dense reconstruction algorithms like Kinect Fusion to acquire dense depth maps. The dense reconstruction methods provides with less noisy and precise depth information, and pose information. We also apply our segmentation framework to other problems like object detection and discovery. We combine the online segmentation approach with object discovery to create offline database of objects thus simplifying the training data acquisition process. We also demonstrate the application of segmentation framework for scalable object detection. We implement the framework on a GPU thus achieving real time performance. Multiview integration and object detection using nearest neighbor is performed on a GPU.

In future, we also plan to handle more complex objects. In this paper we have used a very simple segmentation criterion but some objects do not comply with the convexity criteria for example furniture. However by combining adjacent labels multiple object hypotheses can be generated. These hypotheses can be scored based on prior data to generate scores and select the best one. Also motion is one very important cue which has been successfully used in literature for segmentation of novel objects. In future we plan to to explore motion, combined with depth and other low level cues to identify novel objects and achieve better segmentation. But this will also require more work on dense mapping algorithms as current state of the art works well with no or very less motion.

Bibliography

[1] Loong-Fah Cheong Ajay K. Mishra, Yiannis Aloimonos and Ashraf Kassim. Active visual segmentation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 34(2):639–653, 2012.

[2] Aitor Aldoma, Federico Tombari, Johann Prankl, Andreas Richtsfeld, L. Di Stefano, and Markus Vincze. Multimodal cue integration through hypotheses verification for rgb-d object recognition and 6dof pose estimation. In *ICRA*, pages 2104–2111. IEEE, 2013.

[3] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, May 2011.

[4] Mårten Björkman and Danica Kragic. Active 3d scene segmentation and detection of unknown objects. In *ICRA*, pages 3114–3120, 2010.

[5] Mårten Björkman and Danica Kragic. Active 3d segmentation through fixation of previously unseen objects. In *BMVC*, pages 1–11, 2010.

[6] L. Bo, X. Ren, and D. Fox. Kernel Descriptors for Visual Recognition. In *Advances in Neural Information Processing Systems*, December 2010.

[7] L. Bo, X. Ren, and D. Fox. Depth Kernel Descriptors for Object Recognition. In *IROS*, September 2011.

[8] L. Bo and C. Sminchisescu. Efficient Match Kernel between Sets of Features for Visual Recognition. In *Advances in Neural Information Processing Systems*, December 2009.

[9] Bryan Catanzaro, Bor yiing Su, Narayanan Sundaram, Yunsup Lee, Mark Murphy, and Kurt Keutzer. Efficient, high-quality image contour detection. In *In IEEE International Conference on Computer Vision*, 2009.

[10] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005.

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[12] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, September 2004.

[13] Ross Finman, Thomas Whelan, Michael Kaess, and John J. Leonard. Toward lifelong object segmentation from change detection in dense rgb-d maps. In *ECMR*, pages 178–185. IEEE, 2013.

[14] Aleksey Golovinskiy and Thomas Funkhouser. Min-cut based segmentation of point clouds.

[15] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *CoRR*, abs/1312.6082, 2013.

[16] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *CVPR*, pages 564–571. IEEE, 2013.

[17] Evan Herbst, Peter Henry, and Dieter Fox. Toward Online 3-D Object Segmentation and Mapping. In *International Conference on Robotics and Automation (ICRA)*, 2014.

[18] Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d object discovery via multi-scene analysis. In *IROS*, pages 4850–4856. IEEE, 2011.

[19] Evan Herbst, Xiaofeng Ren, and Dieter Fox. Object segmentation from motion with dense feature matching. In *ICRA Workshop on Semantic Perception, Mapping and Exploration*, 2012.

[20] Steven Hickson, Stan Birchfield, Irfan Essa, and Henrik Christensen. Efficient Hierarchical Graph-Based Segmentation of RGBD Videos. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, June 2014.

[21] Albert S. Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *Int. Symposium on Robotics Research (ISRR)*, Flagstaff, Arizona, USA, Aug. 2011.

[22] Matthew Johnson-Roberson, Jeannette Bohg, Mrten Bjrkman, and Danica Kragic. Attention-based active 3d point cloud segmentation. In *IROS*, pages 1165–1170. IEEE, 2010.

[23] Andrej Karpathy, Stephen Miller, and Li Fei-Fei. "object discovery in 3d scenes via shape analysis". In *"International Conference on Robotics and Automation (ICRA)"*, "2013".

[24] Dov Katz, Moslem Kazemi, J. Andrew (Drew) Bagnell, and Anthony (Tony) Stentz. Clearing a pile of unknown objects using interactive perception. Technical Report CMU-RI-TR-12-34, Robotics Institute, Pittsburgh, PA, November 2012.

[25] Lubor Ladicky, Chris Russell, Pushmeet Kohli, and Philip H. S. Torr. Graph cut based inference with co-occurrence statistics. In *Proceedings of the 11th European Conference on Computer Vision: Part V*, ECCV'10, pages 239–253, Berlin, Heidelberg, 2010. Springer-Verlag.

[26] Lubor Ladicky, Paul Sturgess, Karteek Alahari, Christopher Russell, and Philip H. S. Torr. What, where and how many? combining object detectors and crfs. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *ECCV (4)*, volume 6314 of *Lecture Notes in Computer Science*, pages 424–437. Springer, 2010.

[27] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Detection-based object labeling in 3d scenes. In *IEEE International Conference on on Robotics and Automation*, 2012.

[28] Chaochao Lu and Xiaoou Tang. Surpassing human-level face verification performance on lfw with gaussianface. *CoRR*, abs/1404.3840, 2014.

[29] Ajay K. Mishra, Ashish Shrivastava, and Yiannis Aloimonos. Segmenting "simple" objects using rgb-d. In *ICRA*, pages 4406–4413. IEEE, 2012.

[30] F. Moosmann, O. Pink, and C. Stiller. Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion. In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 215–220, June 2009.

[31] F. Moscheni, S. Bhattacharjee, and M. Kunt. Spatio-temporal segmentation based on region merging. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(9):897–915, Sep 1998.

[32] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, pages 127–136, Washington, DC, USA, 2011. IEEE Computer Society.

[33] B. Rasolzadeh, M. Björkman, K. Huebner, and D. Kragic. An active vision system for detecting, fixating and manipulating objects in the real world. *Int. J. Rob. Res.*, 29(2-3):133–154, February 2010.

[34] X. Ren, L. Bo, and D. Fox. Rgb-(d) scene labeling: Features and algorithms. In *IEEE International Conference on Computer Vision and Pattern Recognition*, June 2012.

[35] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *International Conference on Robotics and Automation*, Shanghai, China, 2011 2011.

[36] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part V*, ECCV'12, pages 746–760, Berlin, Heidelberg, 2012. Springer-Verlag.

[37] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering objects and their location in images. volume 1, pages 370–377, 2005.

[38] Josef Sivic, Bryan C. Russell, Andrew Zisserman, Inria Ecole, and Normale Suprieure. Efros. unsupervised discovery of visual object class hierarchies. In *In Proc. CVPR*, 2008.

[39] Byung soo Kim, Shili Xu, and Silvio Savarese. Accurate localization of 3d objects from rgb-d data using segmentation hypotheses. In *CVPR*, pages 3182–3189. IEEE, 2013.

[40] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. *Conference on Computer Vision and Pattern Recognition (CVPR).*

[41] Antonio Torralba, Rob Fergus, and William T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008.

[42] Ales Ude, Damir Omrcen, and Gordon Cheng. Making object learning and recognition an active process. *I. J. Humanoid Robotics*, 5(2):267–286, 2008.

[43] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. pages 511–518, 2001.

[44] T. Whelan, M. Kaess, M.F. Fallon, H. Johannsson, J.J. Leonard, and J.B. McDonald. Kintinuous: Spatially extended KinectFusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, Australia, Jul 2012.

[45] A. L. Yarbus. *Eye Movements and Vision.* Plenum. New York., 1967.

[46] Edmund Shanming Ye. Object detection in rgb-d indoor scenes. Master's thesis, EECS Department, University of California, Berkeley, Jan 2013.

[47] Dong Zhang, O. Javed, and M. Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 628–635, June 2013.

Appendix A

As discussed before that the performance of proposed method can be improved by incorporating more temporal information. This section presents further proof towards validating this claim.. The following images will demonstrate the performance of proposed method as compared to single frame image segmentation. In order to present the results we will show ground truth segmentation for a particular scene in top left image. In top right panel we will show the segmentation result generated by proposed segmentation framework after 20 frames. In Bottom left panel we will show the results from segmentation performed on the 20th frame. Finally we present a graph in bottom right corner which show the performance improvement as the number of frames increase.

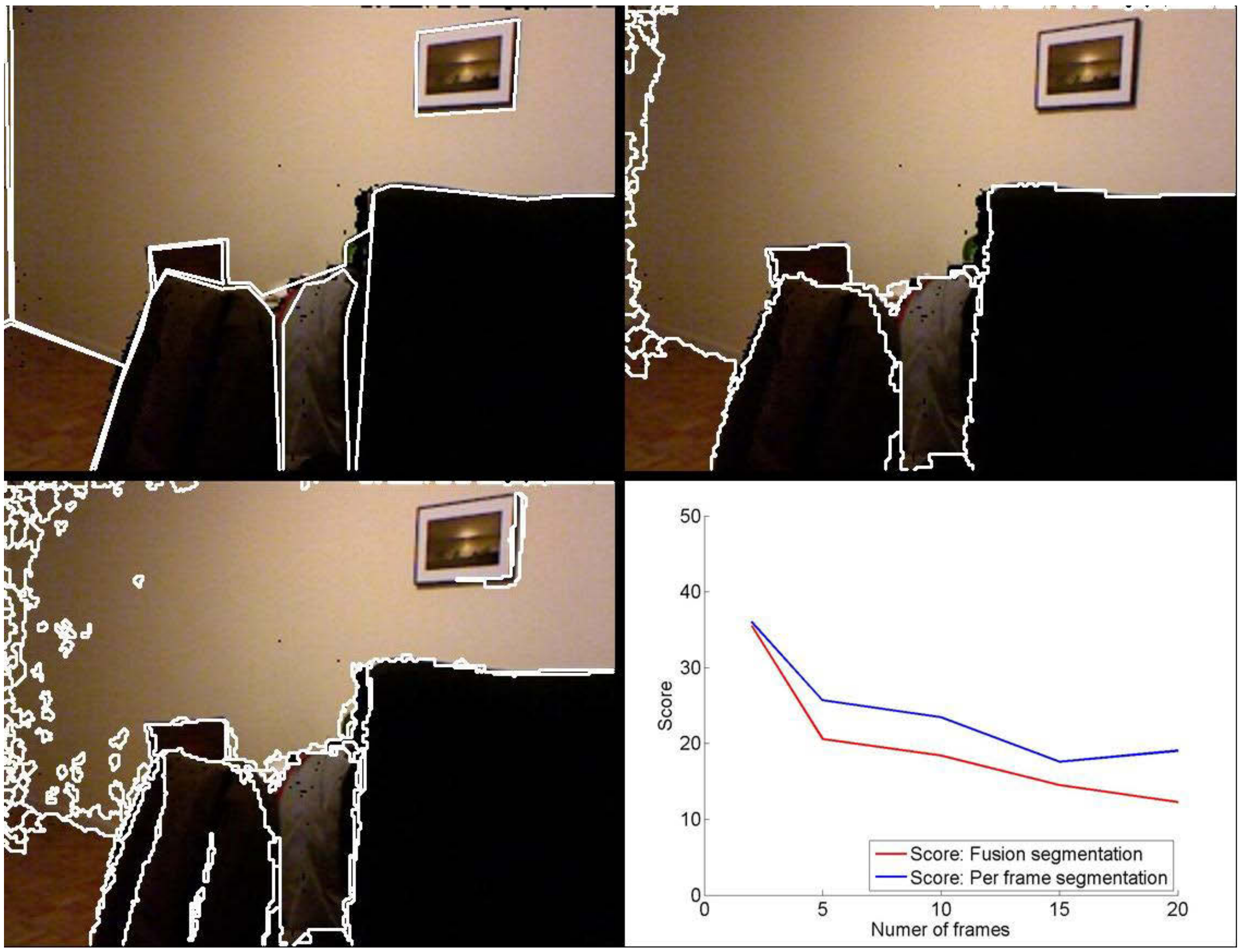


Figure A.1: Top left image: Ground truth object boundaries. Top right: Segmentation results from proposed temporal segmentation Bottom left: Segmentation of single frame. Bottom right: segmentation performance vs number of frames

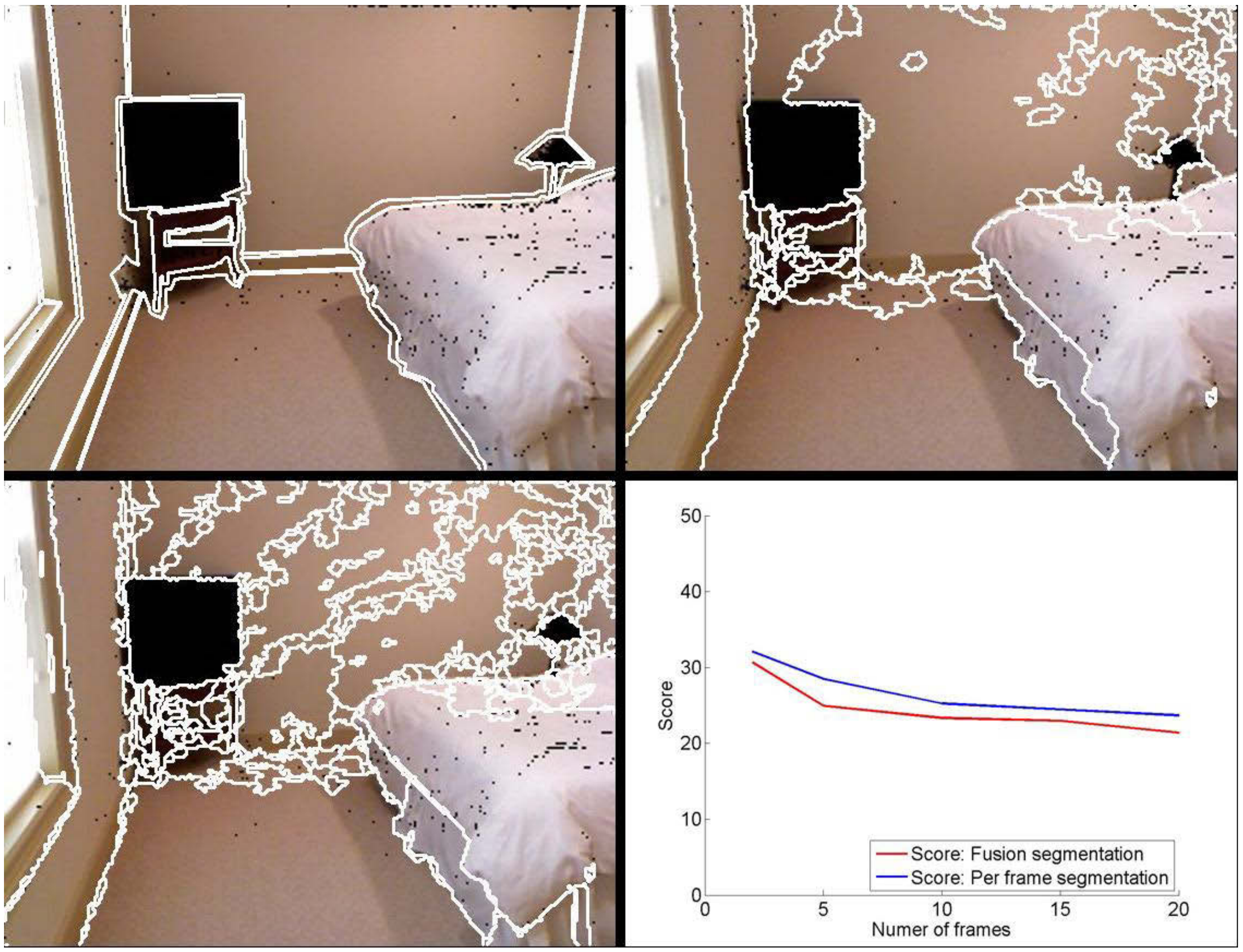


Figure A.2: Top left image: Ground truth object boundaries. Top right: Segmentation results from proposed temporal segmentation Bottom left: Segmentation of single frame. Bottom right: segmentation performance vs number of frames

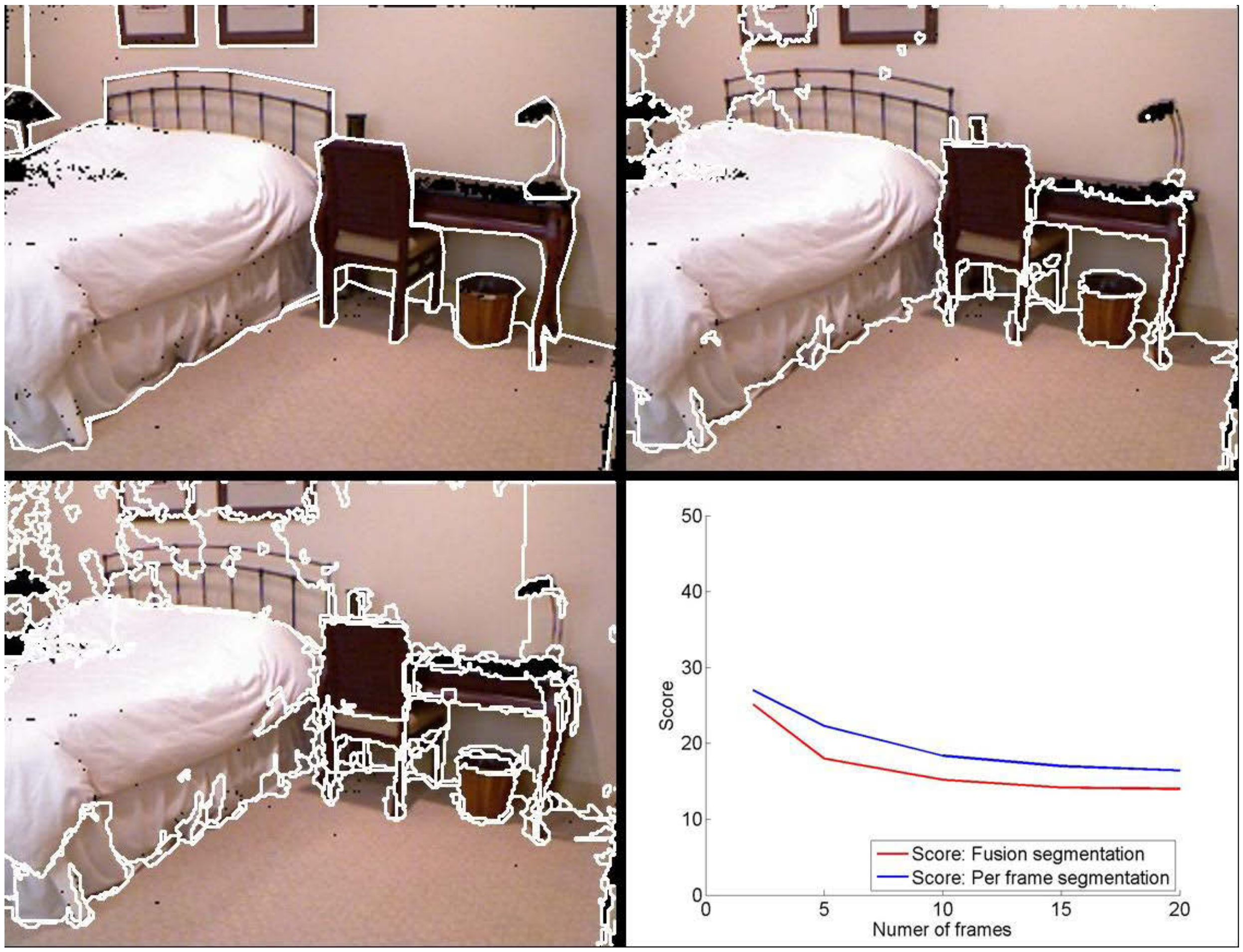


Figure A.3: Top left image: Ground truth object boundaries. Top right: Segmentation results from proposed temporal segmentation Bottom left: Segmentation of single frame. Bottom right: segmentation performance vs number of frames

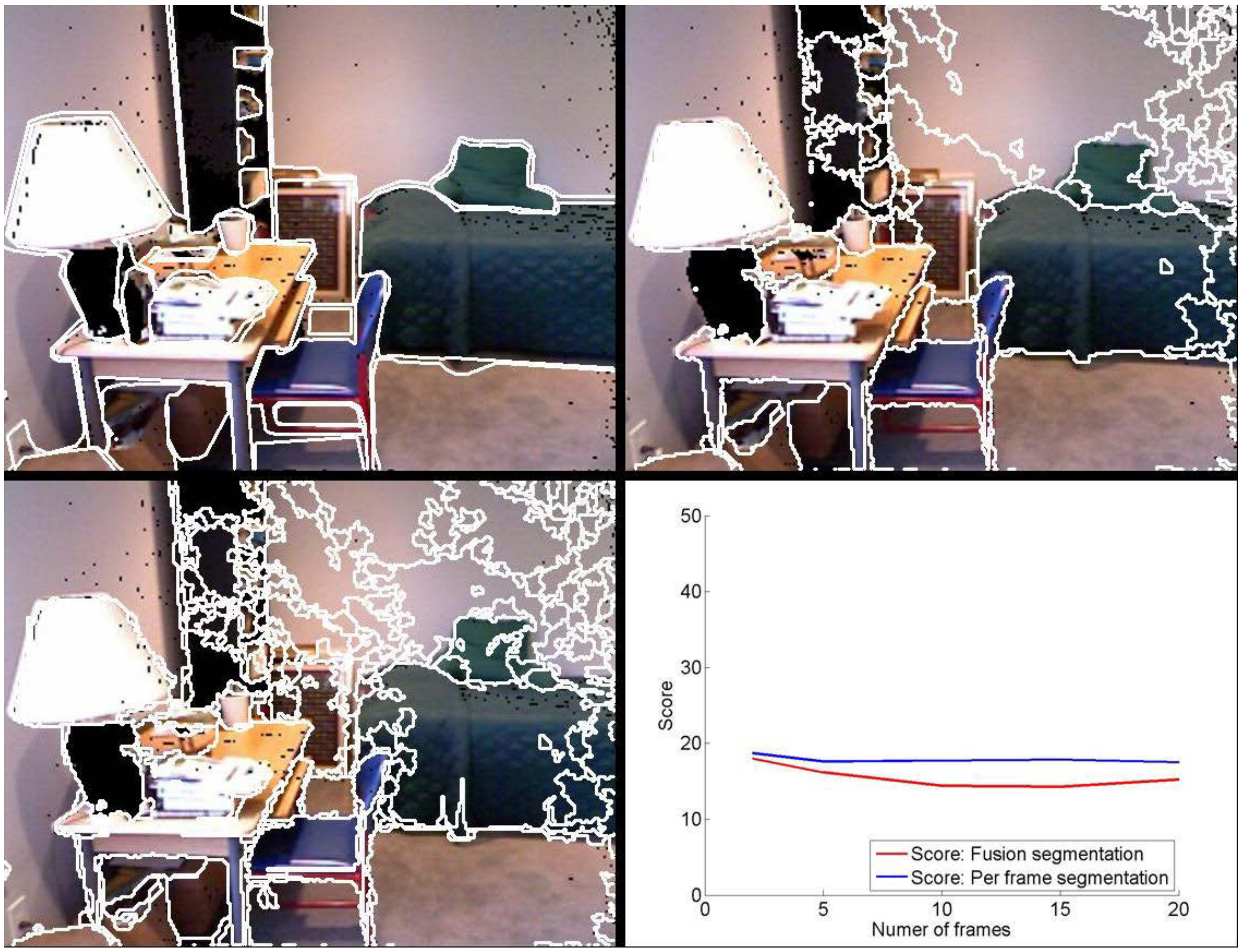


Figure A.4: Top left image: Ground truth object boundaries. Top right: Segmentation results from proposed temporal segmentation Bottom left: Segmentation of single frame. Bottom right: segmentation performance vs number of frames

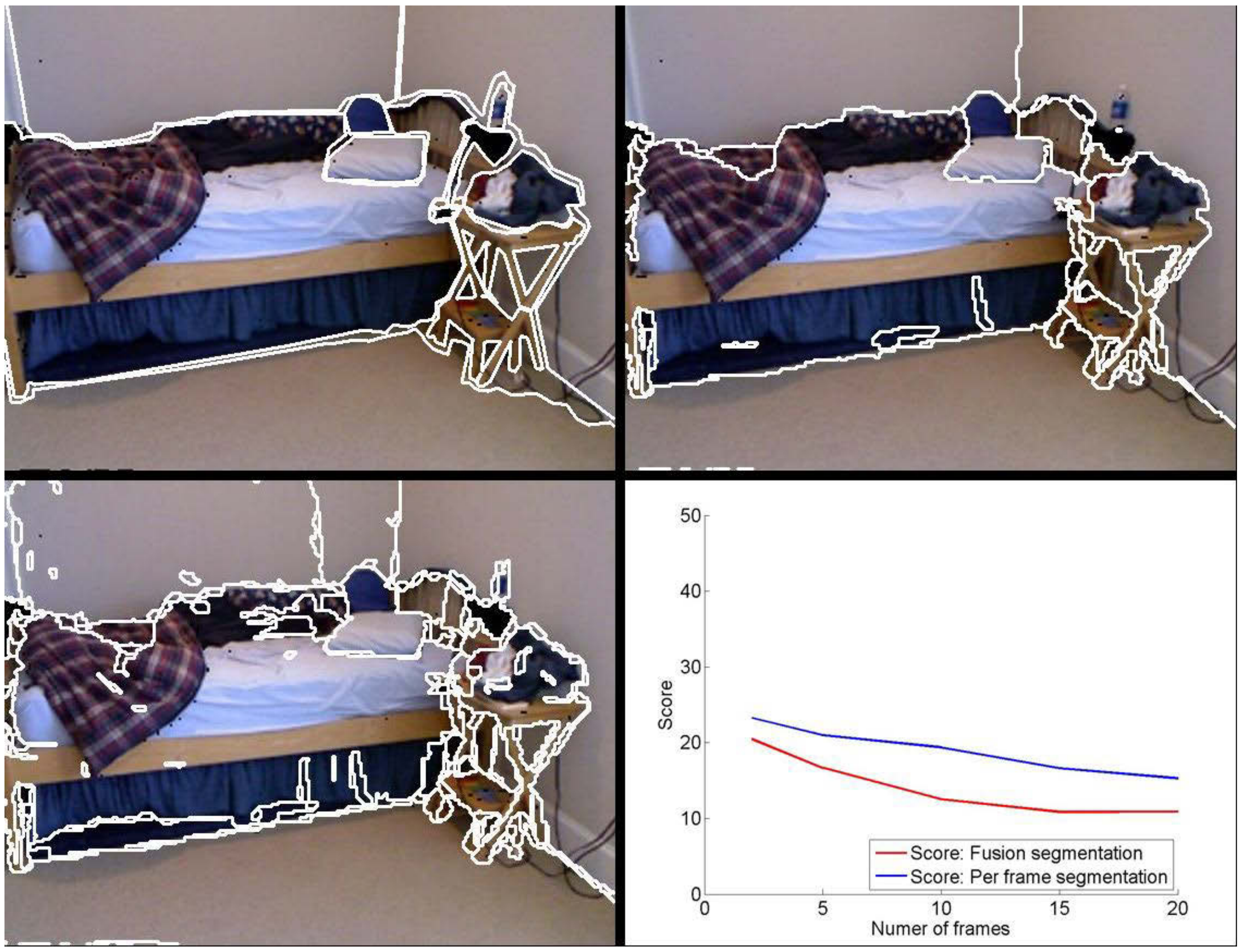


Figure A.5: Top left image: Ground truth object boundaries. Top right: Segmentation results from proposed temporal segmentation Bottom left: Segmentation of single frame. Bottom right: segmentation performance vs number of frames

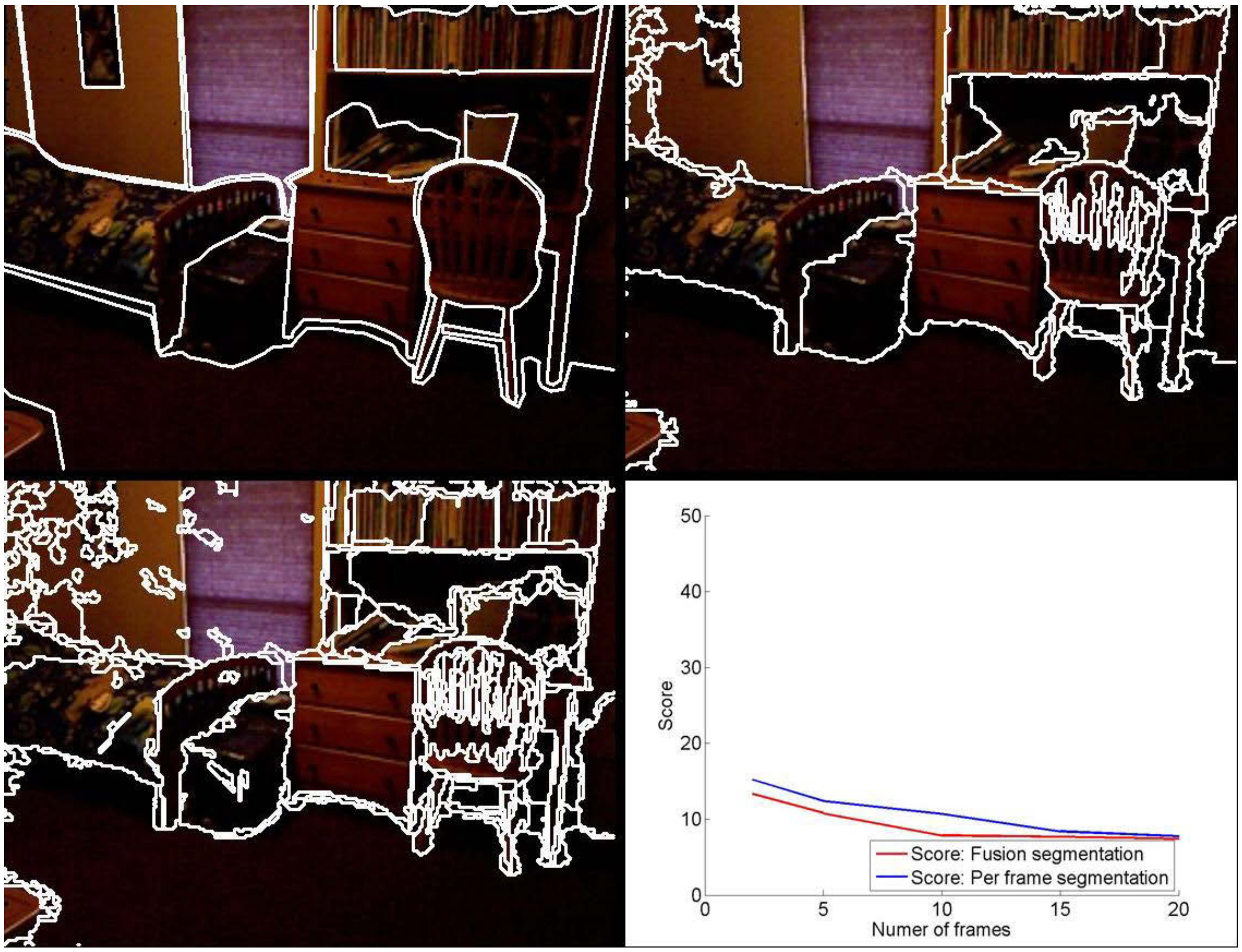


Figure A.6: Top left image: Ground truth object boundaries. Top right: Segmentation results from proposed temporal segmentation Bottom left: Segmentation of single frame. Bottom right: segmentation performance vs number of frames

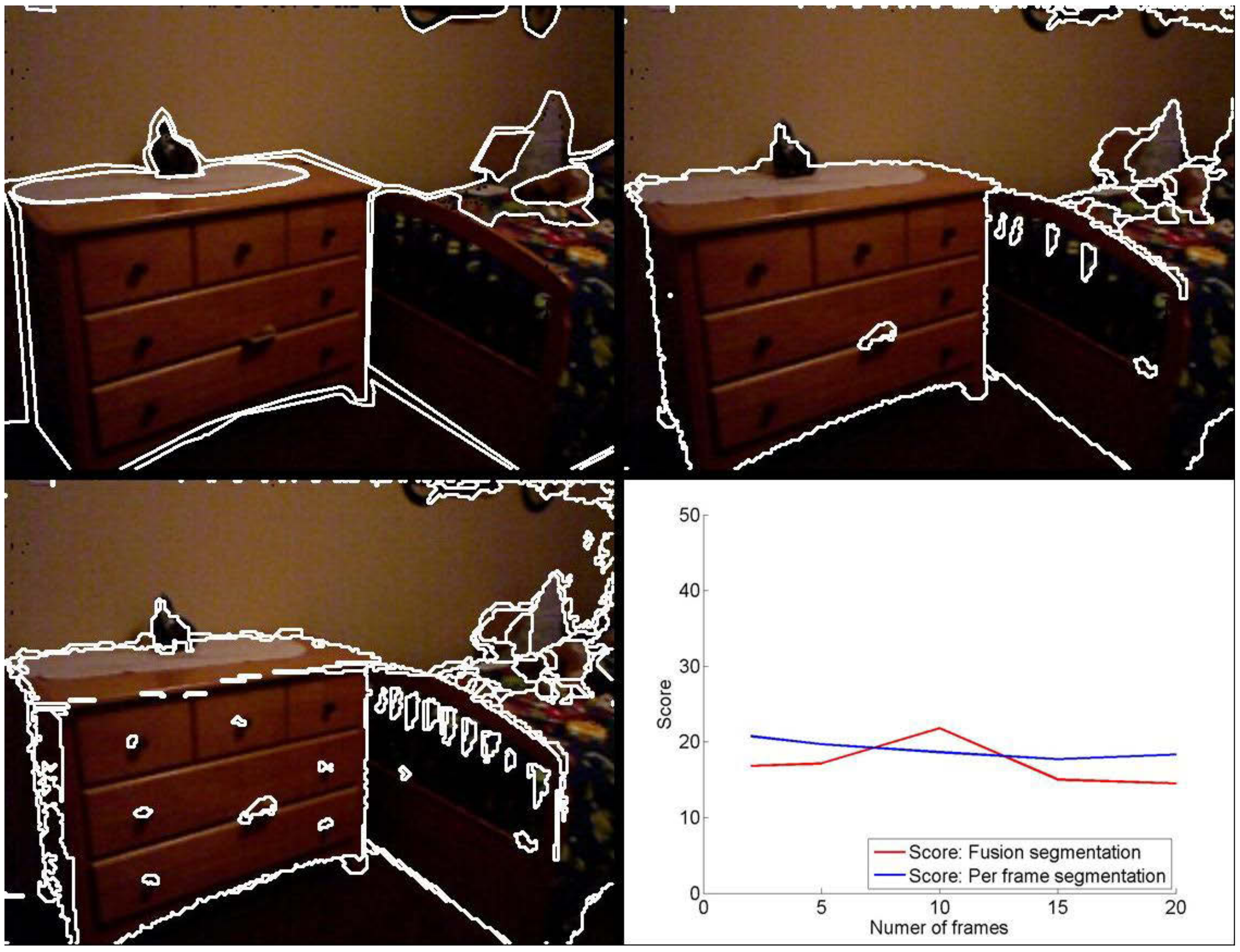


Figure A.7: Top left image: Ground truth object boundaries. Top right: Segmentation results from proposed temporal segmentation Bottom left: Segmentation of single frame. Bottom right: segmentation performance vs number of frames

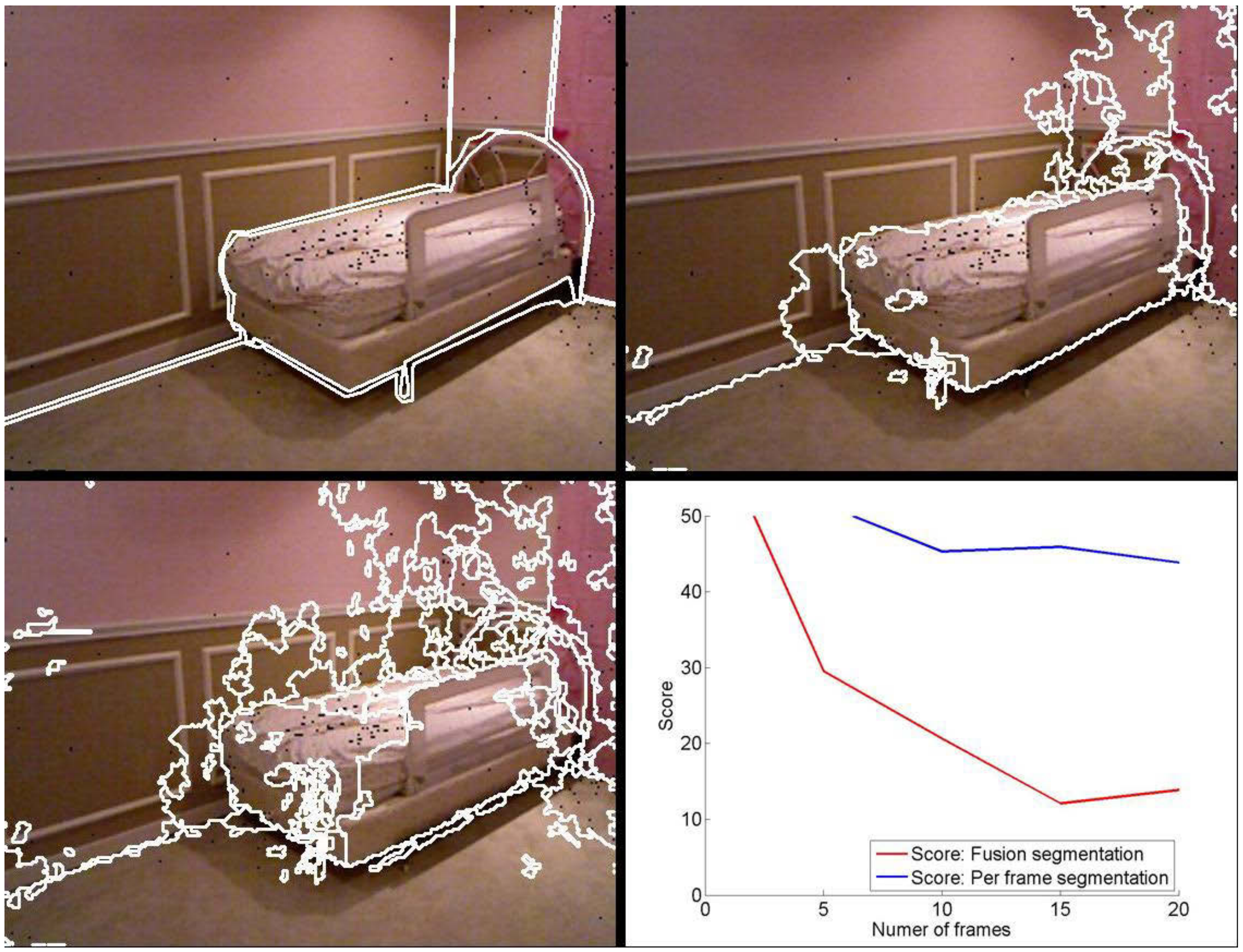


Figure A.8: Top left image: Ground truth object boundaries. Top right: Segmentation results from proposed temporal segmentation Bottom left: Segmentation of single frame. Bottom right: segmentation performance vs number of frames

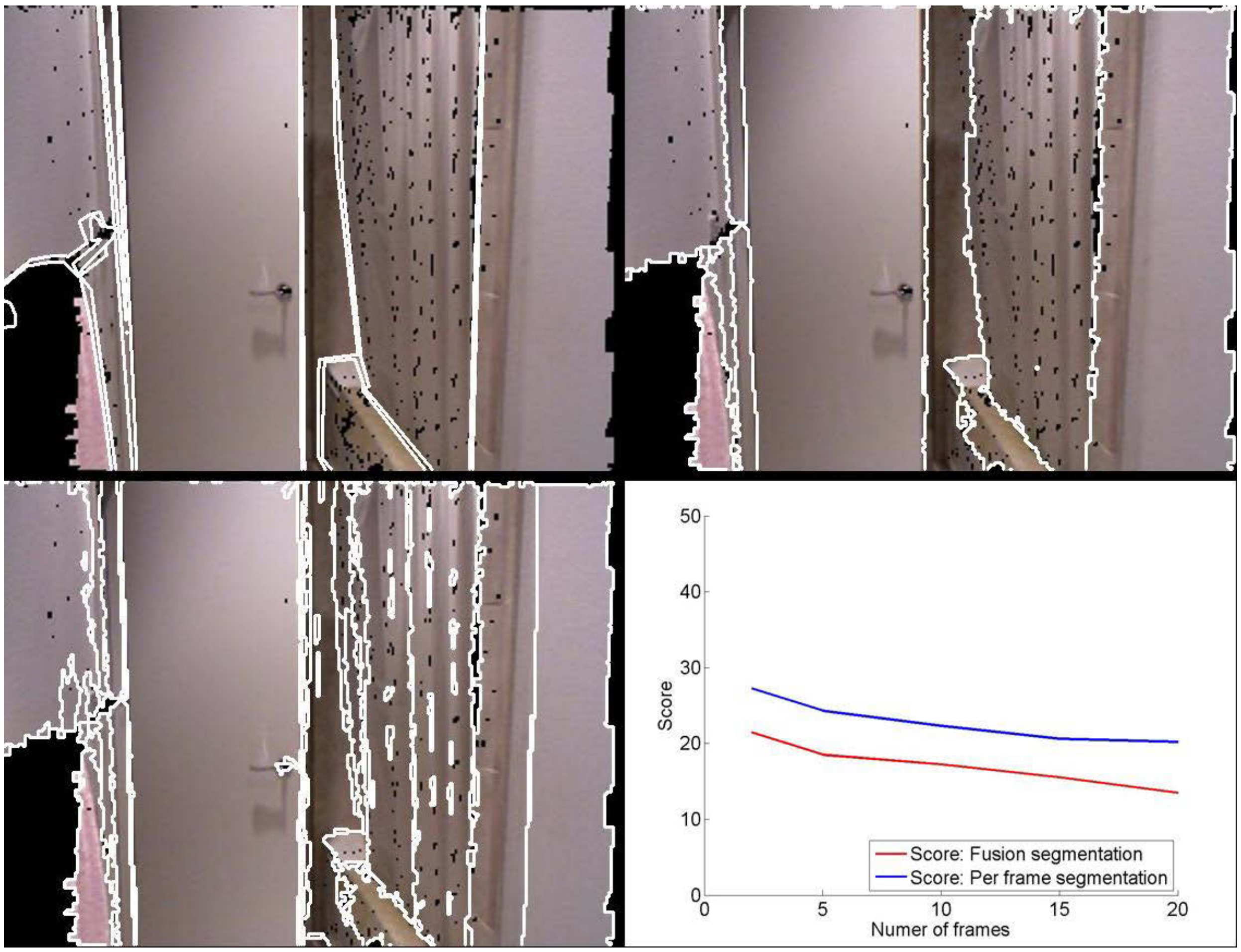


Figure A.9: Top left image: Ground truth object boundaries. Top right: Segmentation results from proposed temporal segmentation Bottom left: Segmentation of single frame. Bottom right: segmentation performance vs number of frames

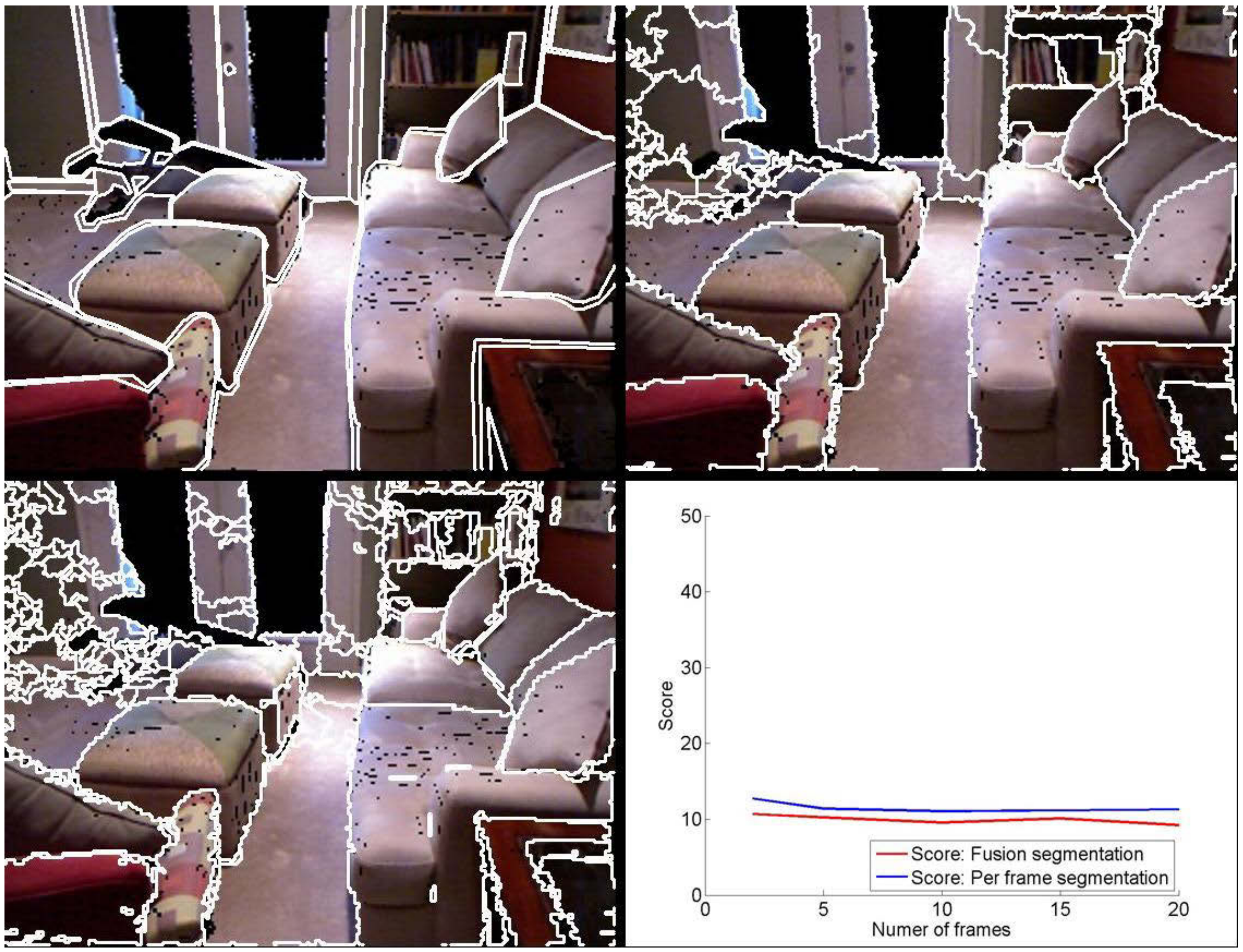


Figure A.10: Top left image: Ground truth object boundaries. Top right: Segmentation results from proposed temporal segmentation Bottom left: Segmentation of single frame. Bottom right: segmentation performance vs number of frames

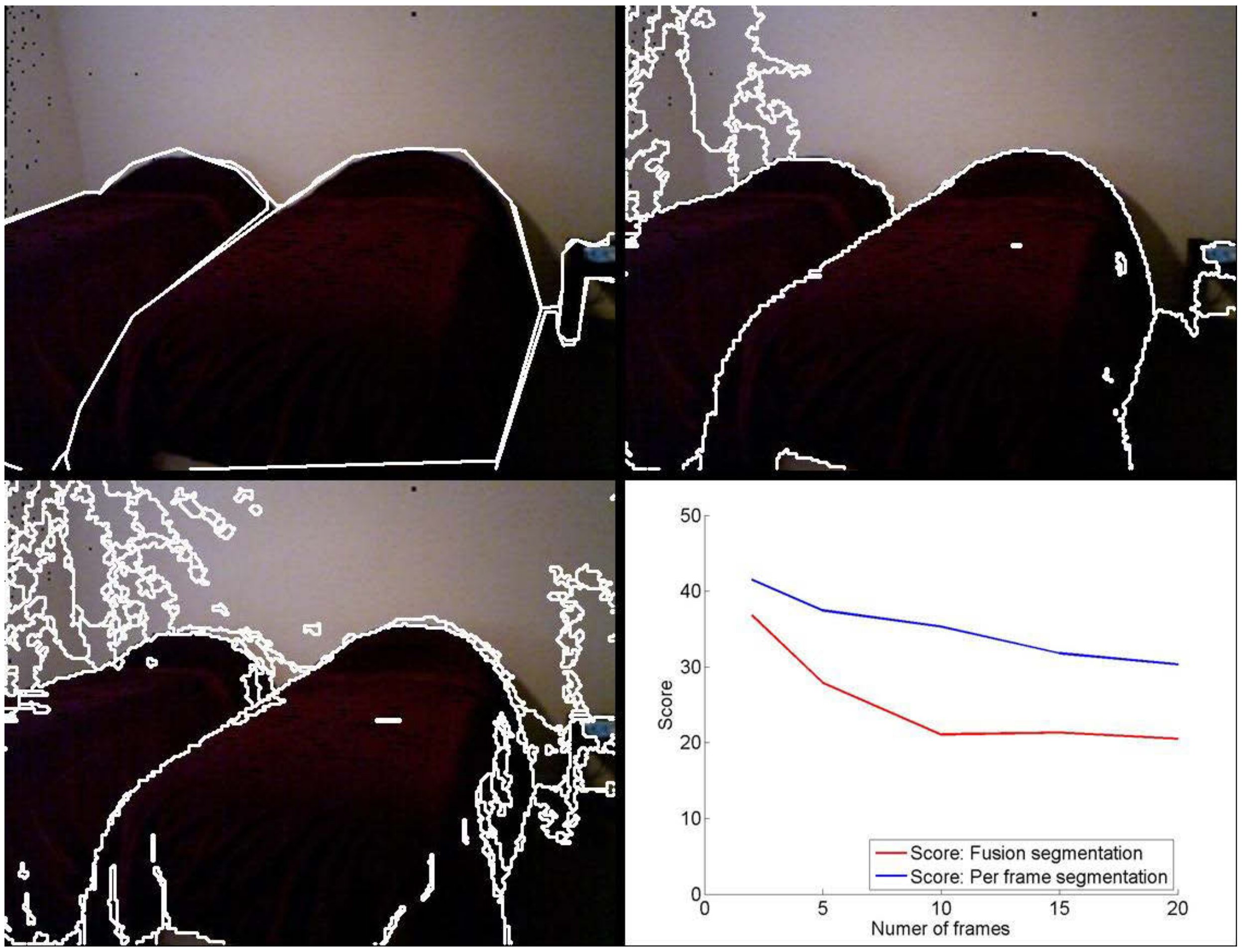


Figure A.11: Top left image: Ground truth object boundaries. Top right: Segmentation results from proposed temporal segmentation Bottom left: Segmentation of single frame. Bottom right: segmentation performance vs number of frames

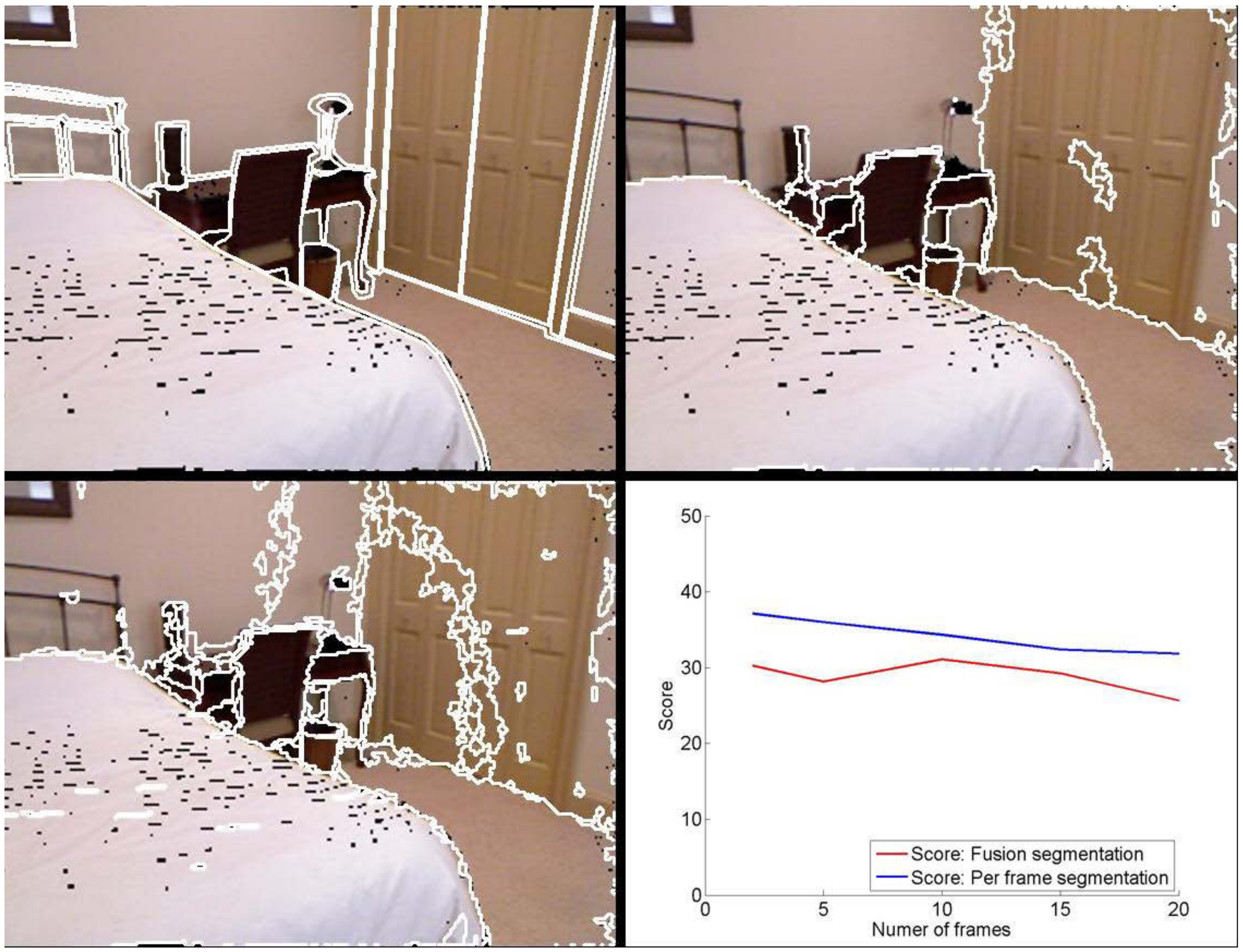


Figure A.12: Top left image: Ground truth object boundaries. Top right: Segmentation results from proposed temporal segmentation Bottom left: Segmentation of single frame. Bottom right: segmentation performance vs number of frames

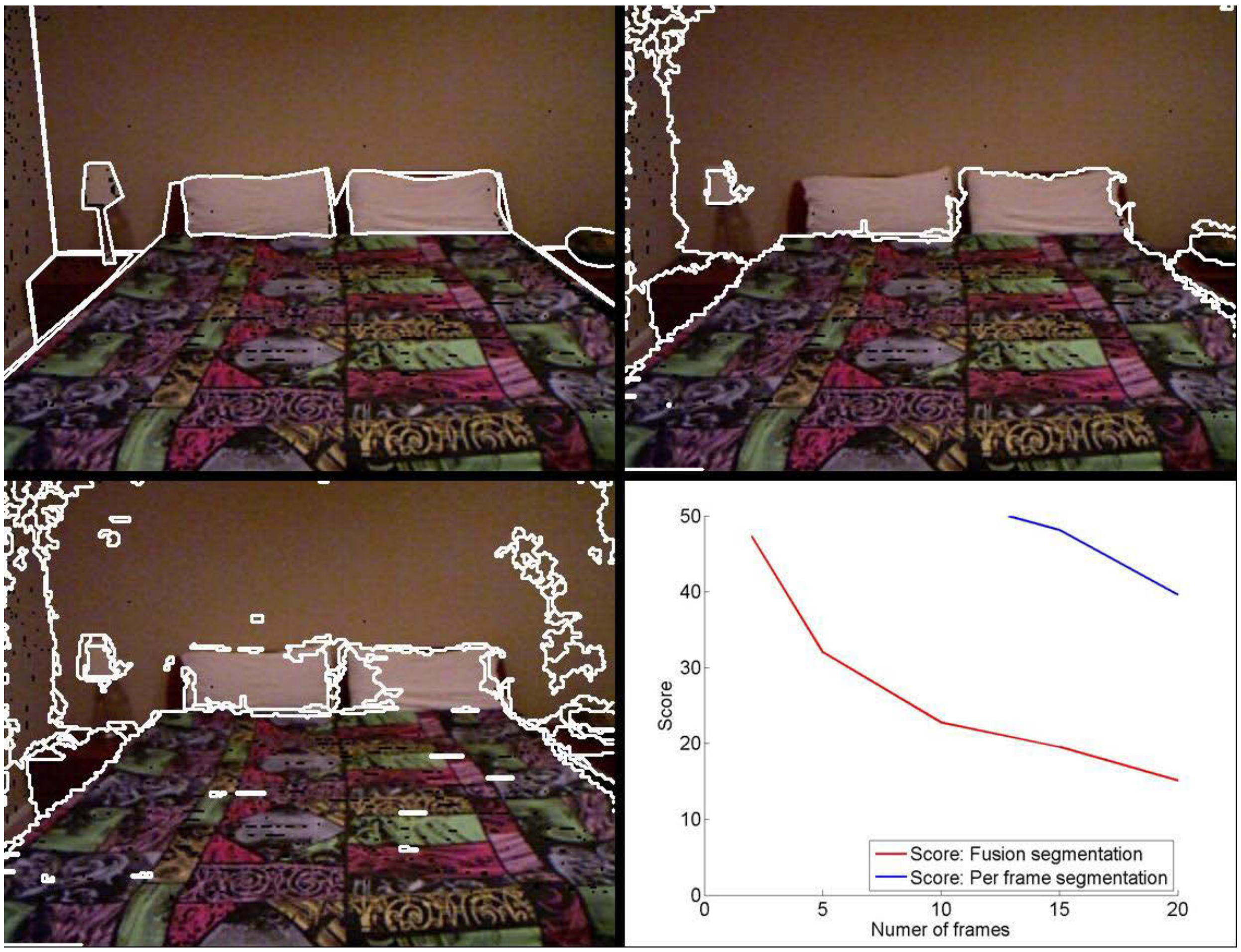


Figure A.13: Top left image: Ground truth object boundaries. Top right: Segmentation results from proposed temporal segmentation Bottom left: Segmentation of single frame. Bottom right: segmentation performance vs number of frames