UNIVERSITY OF GOTHENBURG

The evaluation of different approaches towards using Kinect sensor as a Laser scanner

Bachelor of Science Thesis Software Engineering and Management

KHASHAYAR ABDOLI
ZLATAN HABUL

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, June 2014

The evaluation of different approaches towards using Kinect sensor as a Laser scanner

Khashayar Abdoli
Zlatan Habul

Examiner: Morgan Ericsson

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2014

The Evaluation of Different Approaches Towards Using Kinect Sensor as a Laser Scanner

Khashayar Abdoli
Software Engineering and Management
University of Gothenburg
Gothenburg, Sweden
kh.napster@gmail.com

Zlatan Habul
Software Engineering and Management
University of Gothenburg
Gothenburg, Sweden
zlatan@habul.com

***Abstract*— Transparent objects are easy to recognize with the naked eye, but due to the fact that infrared radiation is travelling through transparent objects, they are not sensed by robots using infrared technology. We propose a workaround to this problem, which improves Simultaneous Localization and Mapping (SLAM) performance. By performing more detailed scans with the Kinect sensor, we are able to find frames around the glass walls, and by that detect them as if they were solid walls. The proposed method is evaluated using the Microsoft Kinect sensor mounted on a Turtlebot robot. Our approach is continuously improved by using the black box software testing method, and we have accomplished good reliability in both the software simulator and the real robot. Results show that our approach gives approximately a 25% more realistic and correct recognition of transparent walls compared to built-in solution in environments that have such walls.**

Keywords— Kinect; Laserscanner; Transparent Walls; Turtlebot; Black-box; Evaluation; depth sensor; ground-truth

I. INTRODUCTION

Localization and mapping are the key requirements in automated self-guiding robots to accomplish navigation. One of the most important techniques in this field is Simultaneous Localization and Mapping (SLAM) [4], where the robots or the autonomous vehicles, while keeping track of their current location, at the same time explore and map the environment around them using sensors data. Traditionally, SLAM is used with a laser scanner together with an odometer. In our experiment we will use a Microsoft Kinect sensor which is a much cheaper device [4].

The Kinect sensor is a motion sensing input device, manufactured by Microsoft for use in the Xbox 360 gaming console, which is shown in Figure 1. Besides being used as a motion sensor, many researchers and developers are using this device in mobile robotics [4, 6, 7, 8, 9]. Compared to the laser scanner, the Kinect sensor has lower accuracy, but it has other advantages. The Kinect sensor measurements are three-dimensional (range and bearing in two directions), while the data from a laser scanner is two-dimensional (range and bearing). This makes the data more information dense [4].

Figure 1. Picture of Microsoft Kinect Sensor [8].

A. *Problem formulation*

The Kinect sensor uses IR radiation, which travels through transparent objects such as glass walls. The result of this is that the robot during SLAM, does not recognize transparent walls and instead it finds the objects behind them. Mapping in such an environment with transparent walls produced by a built-in software solution is therefore inaccurate as we see in Figure 2. Also, because the robot does not recognize transparent walls it can hit them during the navigation. In this paper we will refer to the built-in solution as the original solution.

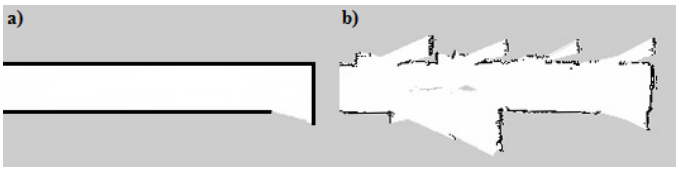

Figure 2. Image a) is bird view drawing of real corridor used for test purposes; most walls are made of glass. Image b) is created by original implementation of ROS during SLAM, as we can see software does not sense glass walls.

In robotic systems with having a many components with rather high complexity it is often hard to dig deep in to components and test the functionality or accuracy of them. Since the Kinect sensor is a small part of the SLAM system in the robot and we wanted to only focus on the accuracy of the Kinect sensor we chose to use a component base black-box testing [17]. Black-box testing is a method of software testing that examines the functionality of an application or module [10], in our case that is a module that converts 3D depth data into 2D data. With using this technique we are able to only focus on the functionality of the Kinect sensor regardless of

the implementation behind it or considering the effects that it can have on the other components. In this paper we have designed an experiment to perfectly suit such test and also measure the accuracy of our results.

B. *Research Goal*

Our main research goal was to evaluate the different approaches that can improve detecting transparent walls, which would give us better mapping of an environment during the SLAM. To be sure that we made improvements, we needed to evaluate both the original solution and the improved solution. For that task we chose Black box testing. Comparing results of the original and improved solution with a real environment gave us an answer. We used the following questions as guidelines for described evaluations:

RQ1: How can a black-box test for SW-components be designed to test the accuracy level of any approach towards using Kinect sensor for mapping system?
RQ2: How accurate are the results from black-box testing for SW-components using full vertical scan in comparison with the original approach which can only scan one height?

C. *Contribution*

- We summarize our main contributions:
- We designed an experiment in a way that black-box software testing would be most useful for evaluating the accuracy of any approach for using Kinect sensor in mapping system.
- We showed that the boundaries of the transparent walls, based on their frames, can be detected with our approach using Kinect sensor.
- We presented data and calculated percentage of improvement comparing to original solution.
- Compared to the original solution, we demonstrated improved mapping with our approach using SLAM.

D. *Structure of article*

The thesis continues with Section 2 that gives information about the Kinect sensor and the original solution. Here we also provide information about improvements in software that we did to be able to evaluate our research. In Section 3, the reader can find information about related work. Methods for data collection are found in Section 4. In Section 5 we present raw data results of our research. Finally the Analysis and Discussion are presented in Section 6 and a conclusion is found in Section 7.

II. BACKGROUND

A. *Turtlebot overview*

Turtlebot is an open-source robot development kit for apps on wheels, which is shown in Figure 3. The system runs on Robot Operating System (ROS) [4], which is a software framework for robot software development. ROS also provides us with a simulator that we have used as a tool for testing of our implementation. With ROS we could build our own map using SLAM and drive the robot through the indoor environment or navigate with a previously recorded map. Besides the wheel encoder, Turtlebot also uses Gyro sensors to calculate Odometry. According to [4], data from the wheel encoder and Gyro sensor is combined in the Kalman filter and the output is the estimated position (Odometry). In this research we focused on improving output from the Kinect sensor, and Odometry is left as it is.

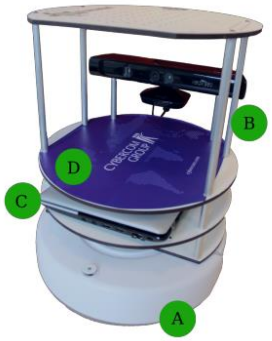

Figure 3. Picture of Turtlebot hardware setup [4], showing a) Creative base unit, b) Kinect sensor mounting, c) Laptop running ROS software and d) Turtlebot module plate.

B. *Kinect's Depth sensor*

In order to implement our approaches, we had to first understand our input data from the Kinect sensor. The Kinect's depth sensor, which was the sensor of interest for our research, provides a 640x480 array of depths which has the view of 58 degrees horizontally and 48 degrees vertically. The firmware of the Kinect sensor performs some initial calculations on the input data. These values are the distance from the sensor in the Z vector which is the direct distance to the object rather the complete distance in a 3D presentation to the camera itself as shown in Figure 4. .

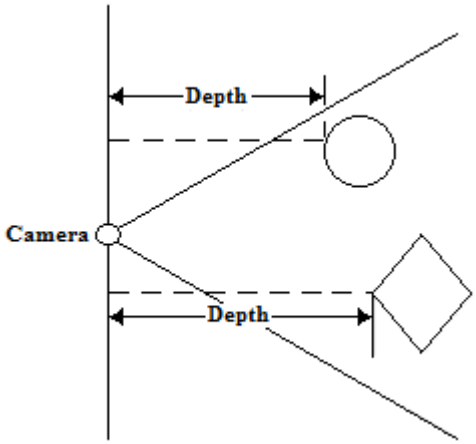

Figure 4. How the depth is calculated in Kinect sensor

In order to use all the data we had to calculate the actual coordinates of the objects detected based on the angle in both axes Y and X. As you can see in Figure 5, these coordinates were defined as follow [7]:

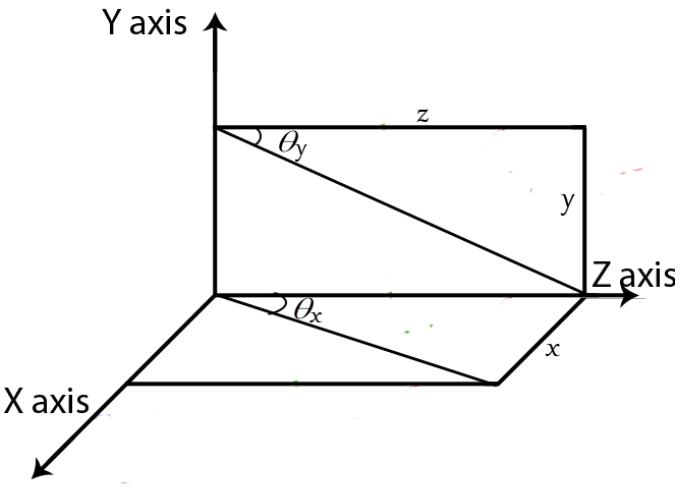

Figure 5. Variables of interest for calculating the position of detected object

$$\theta_x = (column - center_x) * const_x \quad (1)$$

$$\theta_y = (row - center_y) * const_y \quad (2)$$

$$x = \tan(\theta_x) * depth \quad (3)$$

$$y = \tan(\theta_y) * depth \quad (4)$$

$$z = depth \quad (5)$$

For analyzing the Kinect's depth sensor we used MATLAB to visualize the data, and have a full understanding of what they represent. In order to do so, we created a logger system in the robot that recorded the raw data from this sensor, which can be later presented in MATLAB. In Figure 6 a sample of the visualization in MATLAB has been provided.

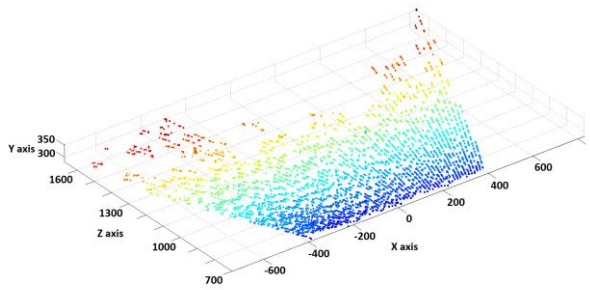

Figure 6 Overview of Kinect sensor depth data in MATLAB

This visualization resulted in understanding that the Kinect's depth sensor was rotated in the simulator and the real robot, since the Kinect was not mounted completely straight on the robot. So we altered the original formulas by adding this rotation value (φ) as defined:

$$y = \tan(\theta_y + \varphi) * depth \quad (4)$$

In Figure 7 original readings are compared to readings after the applied rotation. In this figure all measures are in millimeters.

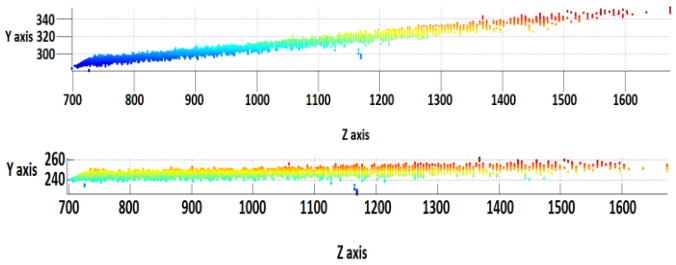

Figure 7 Upper image is showing result without rotation compensation and down image is showing with rotation compensation.

III. IMPLEMENTATION

One of our objectives in this research is that robot will be able to find a transparent wall by finding the frame around the wall. To be able to find a tiny frame, which is only 2 centimeters high, we needed to ensure the Kinect sensor could distinguish precisely between the floor and the frame. Initially, ROS systems were made for laser scanners which output 2D data. Therefore the system has a module that converts 3D depth data into 2D data output, that is compatible with the rest of the system. During this conversion, only the distance of the nearest object at the same vertical line was reported to the system. This module was the subject of all our improvements.

1) Original Solution

The original solution that came with the newly installed ROS system uses a full horizontal scan, but vertically it only scans for ten lines in the middle. With this solution the robot cannot find objects that are much lower or higher than the robot. As shown in the right part (b) of Figure 2 this solution did not detect the frame around the wall.

2) Preparation

First, we logged the height of the floor at all horizontal and vertical lines (640x480) and saved this file as a 2D matrix of the floor height. This logging was done in the open area around the robot so no object would interfere with the floor detection. Analyzing the recorded data by printing results, we found that only 470 vertical lines were containing the data, the other ten lines always had zero value. We also limited the maximum distance to five meters, because according to [8], the precision of Kinect sensor decreases with higher distance. Floor data was found only after 280 vertical lines. This was because the infrared (IR) radiation that scans the middle and upper areas of the environment, which is equal with less than 280 lines, was reaching the five meters limit before reaching the floor.

3) Static Solution

In our first implementation, the floor height was picked up from the log-data file. We called this solution the static solution because floor height was recorded in the file. This means that this solution would not work if we changed Kinect

sensor's height. As stated earlier, the log-data file contained the height of the floor in the 2D matrix. The first dimension was the horizontal position and the second dimension was the vertical position of where the floor was found. When the recorded position matched with the current position of depth data scans from the Kinect sensor, the floor height was picked up from the file and saved as a value. The distance to the objects that were at least 2 centimeters higher than the floor height value were reported. If no object was found, nothing was reported.

4) Dynamic Solution

Our second implementation was based on mathematical formula. We called this solution the dynamic solution because it could be easily adapted to the different heights of the Kinect sensor. After we visualized the log-data file in the MATLAB, we found that our Kinect sensor was not perfectly mounted, it tilted few degrees. We wrote mathematical formula that compensated for the Kinect sensor tilt, and used triangulation to calculate where the floor was during the readings at different vertical and horizontal scans from the Kinect sensor. Results of this mathematical implementation are shown in Figure 8. Our measurement shows that the dynamic solution is better than the static solution with few percent more correct image, compared to the reference image.

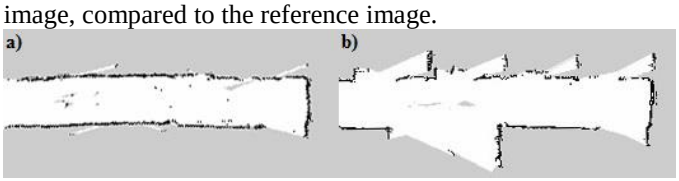

Figure 8. Image a) is created by SLAM with dynamic solution. Image b) is mapping of same corridor with original solution.

IV. RELATED WORK

In this section we have conducted a literature review on earlier publications regarding the use of Kinect sensor in mobile robotics. First we explain how we conducted the search on earlier publications, and then we review them to be able to explain why we cannot use their approach.

A. Databases, strings and keywords

We have chosen to use two databases that by searching gave us relevant publications in our domain. Those are:

- ACM digital library
- IEEE Xplore

Then we chose seven keywords of interest and those are:

- Kinect
- Robot
- Transparent objects
- Wall
- SLAM
- ROS

Our search string is a combination of the chosen words. We had to carefully constructed search strings, otherwise we would find too many papers that had nothing to do with our study. We had to redefine our search strings a few times before we found documents of interest. These are the search strings that we have used;

- Kinect AND robot AND transparent
- SLAM and robot and transparent walls and Kinect
- ROS AND Kinect

B. Results of search:

TABLE 1. Table showing the result of our initial search

	ACM digital	**IEEE Xplore**
Kinect AND robot AND transparent	65	3
SLAM and robot and transparent walls and Kinect	408	96
robot AND localization OR positioning	103	9

C. Inclusion and exclusion criteria

Publications used for this study were selected if they fulfilled certain requirements. We have listed exclusion and inclusion requirements:

Inclusion:

- Publications that have to do with Kinect sensor and transparent objects
- Publications that have to do with Kinect sensor and mobile robotics
- Publication that have to do with SLAM.
- Both Industrial and academic researches and articles.

Exclusion:

- Research papers that use RGB cameras to improve results.
- Research papers that use non-standard devices, like ultra-sonic devices.
- Old research papers from times when computer technology was much slower.
- Non-software related solutions, like mathematical only solutions and others.
- Books, Non-English texts, presentations.

D. Data extraction

Filtering more than 500 research paper that we found by database search, we selected 21 with relevant topic. Reading the abstract of those 21 research papers, we could exclude the ones which did not pass our inclusion and exclusion criteria. At the end, we had five research papers that are relevant and will be used as source to conduct this study. All founded research papers are using other approach than ours.

E. Review

Problem with transparent object and Kinect sensor has been identified before. There has been some attempt to recognize windows, glasses and bottles and recreate them in the 3D map.

Lysenkov et al. [6] proposed a way to recreate transparent object by using Kinect-based detection of no or invalid data in combination with RGB image. They found out that many

TABLE 2. Variables and Parameters

Name of the Variable	Type of the variable	Abbreviation	Type of variable	Scale type	Unit	Range
Scan Range	independent	SR	Internal	ratio	Number	> 0 , < 470
Height of Kinect sensor	independent	Height	Internal	value	Meters	> 0
Rotation of Kinect sensor	independent	Theta	Internal	value	Radians	< PI/2

transparent objects appear as holes in the depth map of Kinect sensor. Those areas they map with RGB object and by that they could recreate 3D image of transparent object. This approach has weakness in the fact that holes can be caused by many other effects. Because even the RGB camera has problem detecting glass walls this approach cannot be used to detect same.

Alt et al. [9] had approach that identifies inconsistent depth measurements of transparent objects while moving Kinect sensor around environment. Those inconsistencies are caused by reflective effects on the surface of object. Detection is limited to transparent object with smooth and curved surfaces like bottles and glasses. Therefore this solution is not suited for our needs.

Kamarudin et al. [7] have proposed a model to convert 3D depth data to 2D map. This research is interesting, because it points out problems with the floor detection, and detection of the transparent object, in their case it was the cabinet's glass door. In their conclusion they stated that detecting transparent and high reflection surfaces still needs to be solved.

Viager [8] has analyzed uses of Kinect sensor for mobile robots. He measured all possible data from Kinect sensor and made more precise and detailed specifications than the one available from official sources. We have used his data in pre-study phase to see if our approach is possible. His conclusion is that Kinect sensor is very viable choice for mobile robot applications, but then he did not analyze what impacts have transparent objects on Kinect sensor readings.

Hjelmare and Rangsjö [4] have carried out a huge amount of work analyzing Turtlebot built-in algorithms for SLAM. They also explain in detail how Turtlebot and all the components that are needed to perform SLAM work, this include Odometry. They had exactly the same hardware as we but their environment was glass-wall free. Most improvement they suggest to Turtlebot had to do with gyro, because they found inaccuracy in map creation when robot is turning. This problem with the Odometry is something that we also have experienced, but it was out of our focus in this research.

V. METHOD

To address RQ1 and RQ2 we designed an experiment according to guidelines provided by [3]. This experiment is described as following;

A. Goals

In order to answer RQ1, Goal 1 was defined as follow:

Goal 1: How can we produce a black-box test that can apply to any approach and validate the out coming results in respect to the ground truth of the map? The focus of this test will be on the input from Kinect sensor and evaluating the output from the map generation program in Turtlebot.

In order to answer RQ2, Goal 2 was defined as follow:

Goal 2: By introducing different scenarios with different amounts of transparent walls, like glass walls, how do the approaches behave and how much of an impact does it have on the accuracy of detection. This will be in respect to detecting all the walls in order to do a complete navigation without hitting objects or walls.

B. Experimental Units

Since the tests are going to be applied to Turtlebot and mainly applied on the Kinect sensor, the Experiment Units are the same as Experiment Materials.

C. Experimental Materials

1) Turtlebot

Our main material to apply the tests was the Turtlebot. With having Turtlebot as an open-hardware solution, it makes it an answer that can be applied to any other indoor robot. Also the map generation system in this robot gives us the opportunity to apply black-box testing and with just applying the approach of concern can get the generated map as an output.

2) Kinect Sensor

Kinect was the essential material for our research where all the approaches affect this sensor of interest. This sensor was mainly chosen because the cost is not high and many different open-source communities work on this sensor, which makes it a general and effective answer as depth sensor.

D. Tasks

In order to carry out the test one must apply the approach to Kinect module of the robot and after finalizing the robot should be put in the environment. For our research we created two sample environments. The first scenario included a solid wall and one transparent wall. In the second scenario, we used a bigger area and included even more glass walls and low height objects. At the end of the tests, we also used the corridor in the company that is mainly constructed with glass wall sections to show the results in the real environment. These three scenarios are included because the amounts of transparent walls are growing in each step and in the end we can see the pattern emerging from these results.

After setting up the environment, the map generation of the robot will run and the output map will be saved as a test case for our black-box testing.

E. Variables and Parameters

The main variables that must be included for our approaches are the range of scan, height of the Kinect sensor, and the rotation of the Kinect sensor. These variables are presented in Table 2.

F. Experiment Design

We conducted our experiment in two phases. First the solutions and behaviors were tested on the simulator designed for the Turtlebot. In the simulator, which simulates the perfect environment with clear and undistorted data, we could ensure that functionality and efficiency of our solution was good. After this stage, we applied the same experiment in the real robot in the environment. In this stage, since the data is not absolute and can be affected by the environment variables such as lightening conditions or going through a non-flat path, we had to ensure the robot could adapt to these conditions.
After this the test was applied to each approach and later on the results compared with each other.

G. Procedure

In order for the experiment to take place first we got familiar with the Kinect's depth sensor. A full understanding was needed in order to be able to apply a full 3D scan and translating that into 2D scan and represent Kinect as a laser scanner.

1) Map Generation

After receiving each of the readings, we generated the map by using the map generation program of Turtlebot. These maps were saved as a binary map file in PGM (Portable Gray Map) format, which black presents the objects and gray or white areas are the possible positions for the robot to navigate to.

2) Generating Ground Truth

For evaluating the extracted maps, we needed the ground truth maps. For generating the Ground Truth [5] in the simulator, we used auto generated data provided by the Turtlebot simulator. In order to generate the Ground Truth in the real environment, we manually created the map by measuring the position of the object in our test scenario. Having generated the ground truth and the maps previously, an accuracy analysis was applied to each approach. In Figure 9 we are showing the generated ground-truth for Small and Big scenario which we generated by measuring the walls manually.

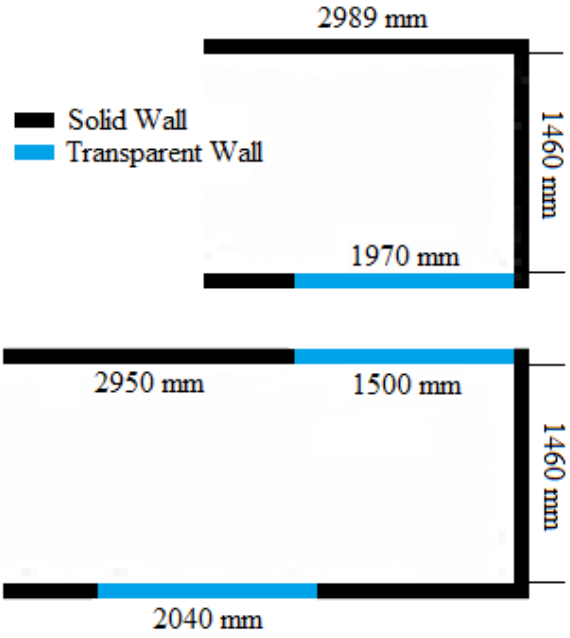

Figure 9. Ground-truth for small and big scenario

H. Analysis

The common measures for evaluating binary maps PASCAL's VOC (Visual Object Classes) [13, 11, 2] and F_β-measure [16, 14, 12, 18, 19]. We also used an improved version of the $F_\beta - measure$ called Weighted F_β-measure ($F_\beta^\omega - measure$) [15].

All of these tests use four concepts for their evaluation:

- True Positive (TP): the quantity of the points that are true and they refer to a true value on the ground truth.
- True Negative (TN): the quantity of the points that are false and they refer to a false value on the ground truth.
- False Positive (FP): the quantity of the points that are true and they refer to a false value on the ground truth.
- False Negative (FN): the quantity of the points that are false and they refer to a true value on the ground truth.

By use of these values each test calculates the accuracy of the map.

1) PASCAL's VOC

This test is one of the basic tests that just calculates the ratio between true-positives and the sum of true-positives, false-negatives and false-positives.

$$PASCAL'sVOC = \frac{TP}{TP + FN + FP} \quad (6)$$

This test is considered as a very strict test since it only focuses on the points that are exactly correct reading.

2) $F_\beta - measure$

This test uses two of most common qualities, which are Recall and Precision:

$$Recall = \frac{TP}{TP + FN} \quad (7)$$

$$Precision = \frac{TP}{TP + FP} \quad (8)$$

And based on these values the test will was calculated as follows:

$$F_\beta - measure = (1+\beta^2)\frac{Precision \times Recall}{\beta^2 \times Precision + Recall} = \frac{(1+\beta^2) \times TP}{(1+\beta^2) \times TP + \beta^2 \times FN + FP} \quad (9)$$

Where β is a control variable that controls the preference between over-detection and complete-detection, which is typically one and we will use one in our research as well. This test focuses more on the advantages of the picture rather than the failures, so this test is not as strict as PASCAL's VOC test.

3) $F_\beta^\omega - measure$

For this test we are using the program based on the paper "How to Evaluate Foreground Maps" [13], which is provided by the authors. This measure as stated in the paper is calculated as below:

$$F_\beta^\omega - measure = (1+\beta^2)\frac{Precision^\omega \times Recall^\omega}{\beta^2 \times Precision^\omega + Recall^\omega} \quad (10)$$

This test is performed by checking the neighbors of the reading, and comparing the result more in terms of the shape rather than exact positions. This is a better way to understand if the readings are referring to the ground truth.

VI. RESULTS

A. Map

Based on the maps extracted from Turtlebot the Ground Truth [5] for each of scenario will be created.

1) Small Scenario

In Figure 10 the small scenario results are presented, the generated ground-truth on the top-left-corner, the original solution is on the top-right-corner, the static solution is on the bottom-left-corner and the dynamic solution is presented in the bottom-right-corner.

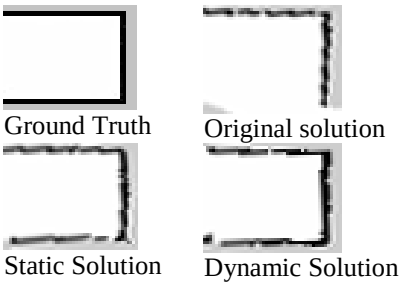

Figure 10. Ground-truth and extracted map for small scenario

2) Big Scenario

In Figure 11 the big scenario results are presented, the generated ground-truth is on the top-left-corner, the original solution is on the top-right-corner, the static solution is on the bottom-left-corner and the dynamic solution is presented in the bottom-right-corner.

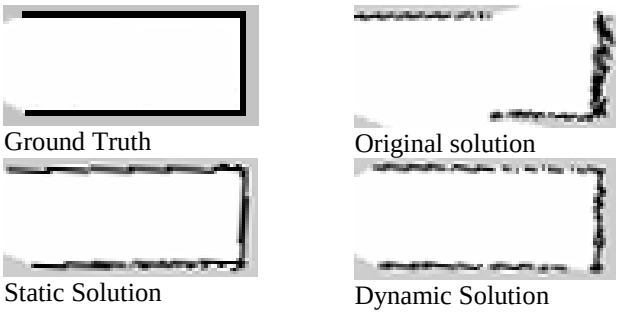

Figure 11. Ground-truth and extracted map for big scenario

3) Corridor Scenario

In Figure 12 the corridor scenario results are presented. The generated ground-truth is on the top-left-corner, the original solution is on the top-right-corner, the static solution is on the bottom-left-corner and the dynamic solution is presented in the bottom-right-corner.

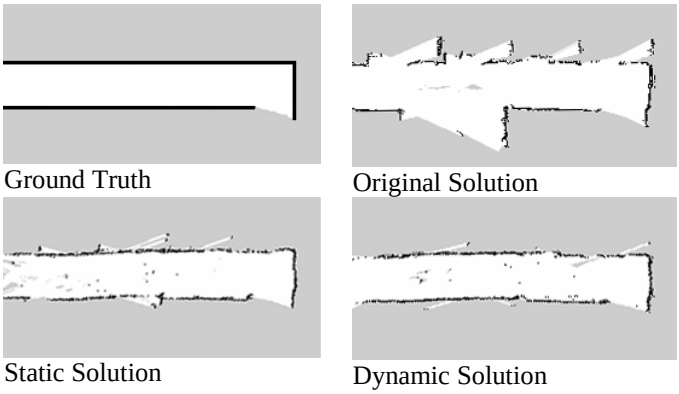

Figure 12. Ground-truth and extracted map for corridor scenario

B. Evaluation

We have applied our three test approaches to each of the scenarios and the results are presented in column-charts.

1) PASCAL's VOC

After applying this test to our extracted maps based on the ground truth in the small scenario the result we got was:

- Original solution was 40.55%
- Static solution was 50.71%
- Dynamic solution was 64.25%.

In the big scenario the result we got was:

- Original solution was 33.80%
- Static solution was 53.34%
- Dynamic solution was 62.37%.

And applying the same test in the corridor of the company we got these results:

- Original solution was 12.05%
- Static solution was 47.76%
- Dynamic solution was 59.91%.

The results in comparison with each other are presented in Figure 13.

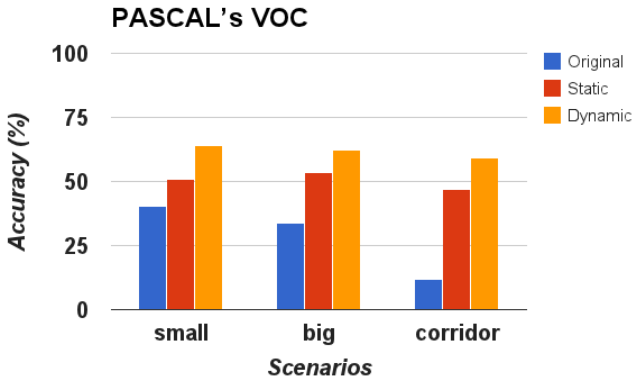

Figure 13 Accuracy test for PASCAL's VOC

2) $F_\beta - measure$

After applying this test to our extracted maps based on the ground truth in the small scenario the result we got was:

- Original solution was 57.7%
- Static solution was 67.28%
- Dynamic solution was 78.24%.

In the big scenario the result we got was:

- Original solution was 50.63%
- Static solution was 69.57%
- Dynamic solution was 73.07%.

And applying the same test in the corridor of the company we got these results:

- Original solution was 22.12%
- Static solution was 57.86%
- Dynamic solution was 71.48%.

The results in comparison with each other are presented in Figure 14.

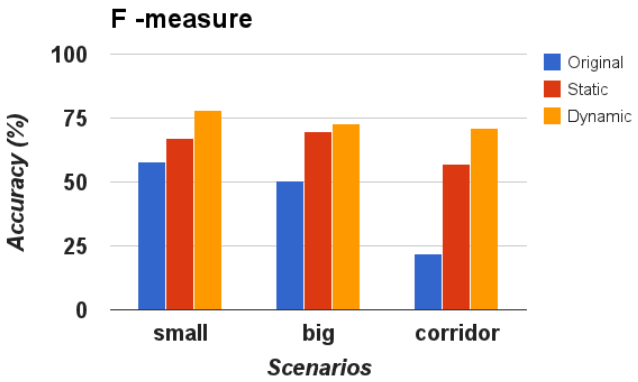

Figure 14 Accuracy test for F-measure

3) $F_\beta^\omega - measure$

After applying this test to our extracted maps based on the ground truth in the small scenario the result we got was:

- Original solution was 67.12%
- Static solution was 76.26%
- Dynamic solution was 84.71%.

In the big scenario the result we got was:

- Original solution was 60.39%
- Static solution was 80.65%
- Dynamic solution was 85.03%.

And applying the same test in the corridor of the company we got these results:

- Original solution was 26.93%
- Static solution was 73.28%
- Dynamic solution was 81.83%.

The results in comparison with each other are presented in Figure 15.

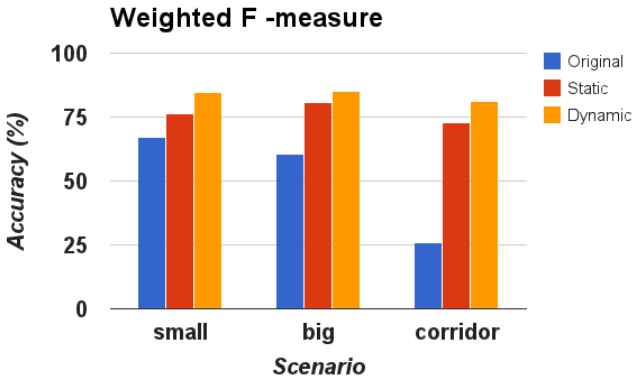

Figure 15 Accuracy test for Weighted F-measure

VII. Discussion

A. *Evaluation of Results and Implications*

Based on the results extracted from our three tests, we can clearly see that the dynamic solution has a clear advantage over the original solution. The accuracy level compare has improved at least 23% and this increase up to 45% improvement.

One of the important things to mention is that by introducing more transparent walls in each scenario, the accuracy of the original solutions drops, but in the dynamic solution we can see a consistency throughout all the scenarios.

To compare the static solution and dynamic solution we must go back and compare the procedure to which they can be applied. In the static solution, to obtain a clear reading, we must first calibrate the robot in an open area so that the readings for the floor can be recorded. Later on if any changes are applied to the robot, this action must be repeated, whereas in the dynamic solution no calibration phase is needed. By changing the height value in the solution and setting in the actual height of the Kinect sensor, the approach can be applied.

In addition to the phase in all scenarios with all the tests, the dynamic solution shows better results. This is due to the fact that the calibration can be really sensitive, and the readings for all points are not available at all times.

By using black-box testing in our research, we have shown that these tests can be applied to any given approach for scanning the area and without knowing how the approach is working a clear accuracy level will be generated which can be later used for comparison between other approaches. In complex systems such as robot operating systems (ROS), it is usually very hard to understand all components and dig into the system to understand the behaviors. It is also difficult to understand how the different approaches affect the results, but with using black-box testing and using tests such as PASCAL's VOC, F-measure and Weighted F-measure, one can simply extract the map in any way and by applying these tests, understand if the approach is good or not.

B. Threats to Validity

1) Threats to construct validity

One of the main threats to our construct is applying the tests. PASCAL's VOC, F-measure and Weighted F-measure all need to have the inputs exactly the same, meaning that the extracted maps must be exactly the same size and orientation. This is due to the fact that true-positives and similar concept compare the point to the corresponding point in the ground-truth, and if this requirement is not met the results will not be valid.

2) Threats to internal validity

The main threat to our internal validity is hardware failures. If the Kinect sensor is in an environment where it will receive direct light, the results will be corrupted. Even in normal conditions, the sensor can still lose a lot of data [1].

Also the error in the Odometry can be a problem during the map generation process. If the Odometry fails, the resulting map will be rotated more or less than it should, which can result in an inaccurate map, which is not cause by any of the approaches related to using the depth sensor. The Odometry error was discovered when we conducted mapping of the L shaped corridor. As stated before, Odometry was not the aim of this research so we did not calculate the Odometry error. To avoid this error, in our experimental setting, we chose not to make any turns while we collected data from the Kinect sensor.

3) Threats to external validity

In our solutions in order to detect the transparent walls we were looking for the frame of the walls and detect them and then realizing the wall. If in an environment a scenario is introduced that the transparent wall have no frame located on the ground or in the view of the robot then our solutions might fail. This threat can be more generalize to any solution since the depth sensor would not be able to detect such case and maybe solutions like using RGB cameras might be of help.

VIII. CONCLUSION

A. Summary

Localization and mapping are the key requirements in automated self-guiding robots to accomplish navigation. One of the most important techniques in this field is Simultaneous Localization and Mapping (SLAM). SLAM is processed in our test robot by outputs from Kinect sensor and Odometry. We have been focused to evaluated three different approaches of using Kinect sensor in mobile robotic. Aim is to find approach that makes mapping more reliable in indoor environments with transparent walls. We have used the black box software testing method to test functionality of our improved implementation. Furthermore, test results, which are presented as images of detected environment, was evaluated using three different accuracy tests; PASCAL's VOC, F-measure and Weighted F-measure.

B. Impact

Original built-in solution works good in the environments without transparent walls, however as expected based on technology of Kinect sensor this solution shows leak of robustness in the environment with the transparent walls. Having that in mind we have developed two new approaches that we call Static and Dynamic approach. Both of those approaches shows improvement comparing to the original solution and can be used in environments that have frame around transparent walls. In our knowledge there is no environment that does not have solid frame around transparent walls. Our best solution Dynamic approach, which is called by that because it can be adapted to other robots and environments, is showing approximately 25% better results than original solution. This percentage would be even higher if our test scenario had only transparent walls but in that case original solution would totally fail and we do not find realistic that some indoor environment only have transparent walls. If we look at picture Figure 11 in part which shows corridor under Dynamic solution, we can see that transparent walls are detected totally. If we feed this picture to robot operating system (ROS) and make robot navigate to place behind transparent walls it will stop before reaching wall, same test done by picture made by original solution will make robot hit the wall. With those simple tests we have shown that our Dynamic approach is considered as a very viable choice for use with mobile robots in indoor environments.

C. Future Work

We have shown that our approaches work with the Microsoft Kinect sensor in the environments with transparent walls, by finding the frame around the wall. However, when there is no frame around the transparent wall this solution will not give any improvements. We believe there are two possibilities for future work in this area; adding an ultra-sonic sensor on the mobile robot, or further processing methods that compare depth scans in relation to RGB picture and in that way somehow find transparent walls. Future work which researches a unit that handles Odometry would be also a good subject for black box testing, as we noticed possible imprecision in calculating how much the robot turned.

ACKNOWLEDGMENT

We would like to thank Tom Strömberg and Hans Forsberg at Cybercom Group for giving us opportunity to carry out our thesis work at Cybercom, and for lending us a robot to perform different experiments. We would also like to thank Ana Magazinius at University of Gothenburg for giving us guidelines and for the material she provided us. Finally we would like to thank our supervisor Christian Berger at University of Gothenburg who pointed us in the right direction, which helped us writing this paper.

REFERENCES

[1] A. Dakkak, A. Husain, "Recovering Missing Depth Information from Microsoft's Kinect", unpublished.

[2] A. Joulin, F. Bach, and J. Ponce, "Multi-class cosegmentation", In CVPR, pp. 542–549, 2012.

[3] F. Shull, J. Singer, D.I.K. Sjøberg, "Guide to Advanced Empirical Software Engineering", Springer-Verlag New York, Inc. Secaucus, NJ, USA,pp.201-221, 2007.

[4] F. Hjelmare, J. Rangsjö "Simultaneous Localization And Mapping Using a Kinect in a Sparse Feature Indoor Environment", Linköpings universitet, Linkoping,Sweden, 2012.

[5] G. Zi, "Groundtruth generation and document image degradation", Language and Media Processing LaboratoryInstitute for Advanced Computer Studies University of Maryland, May 2005.

[6] I. Lysenkov, V. Eruhimov, and G. Bradski, "Recognition and pose estimation of rigid transparent objects with Kinect sensor," in Proc. of Robotics: Science and Systems, Sydney, Australia, July 2012.

[7] K. Kamarudin, S.M. Mamduh, A.Y.M. Shakaff, S.M. Saad, A. Zakaria, A.H. Abdullah, L.M Kamarudin, "Method to convert Kinect's 3D depth data to a 2D map for indoor SLAM" Centre of Excellence for Adv. Sensor Technol. (CEASTech), Univ. Malaysia Perlis, Arau, Malaysia, March 2013.

[8] M.Viager, "Analysis of Kinect for Mobile Robots", DTU Electrical Engineering, Technical University of Denmark, Individual course report, Denmark, March 2011.

[9] N. Alt, P. Rives, E. Steinbach "Reconstruction of transparent object in unstructured 3D scenes with depth camera", Inst. for Media Technol., Tech. Univ. Munchen, Munich, Germany, Sept. 2013.

[10] N. Sirohi, A. Parashar, "Component Based System and Testing Techniques", International Journal of Advanced Research in Computer and Communication Engineering,Vol. 2, Issue 6, June 2013.

[11] M. Everingham, L. Van Gool, C. K. I.Williams, J.Winn, and A. Zisserman. "The Pascal Visual Object Classes (VOC) Challenge", In IJCV, 88(2), pp. 303–338, 2010.

[12] M.M. Cheng, G.X. Zhang, N. J. Mitra, X. Huang, and S.M. Hu, "Global contrast based salient region detection", In CVPR, pp. 409–416, 2011.

[13] P. Arbelaez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik. "Semantic segmentation using regions and parts",In CVPR, pp. 3378–3385, 2012.

[14] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. "Contour detectionand hierarchical image segmentation", In PAMI, 33(5), pp. 898-916, 2011.

[15] R. Margolin, L. Zelnik-Manor, A. Tal, "How to Evaluate Foreground Maps?", In CVPR 2014.

[16] S. Alpert, M. Galun, R. Basri, and A. Brandt, "Image segmentation by probabilistic bottom-up aggregation and cue integration", In CVPR, pp. 1–8, June 2007.

[17] S.U. Rehman, F. Naseer, K. Hussain, "Using meta-data technique for component based black box testing", Emerging Technologies (ICET), 2010 6th International Conference, Univ. of Arid Agric., Inst. of Inf. Technol., Rawalpindi, Pakistan. October 2010.

[18] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, and H. Shum, "Learning to detect a salient object", In PAMI, pp.1–8, 2010.

[19] X. Shen and Y. Wu, "A unified approach to salient object detection via low rank matrix recovery", In CVPR, pp. 853-860, 2012.