

Ben-Gurion University of the Negev  
Faculty of Engineering Sciences  
Department of Industrial Engineering and  
Management

**Development of algorithms for a  
human-following robot equipped  
with Kinect vision and laser sensors  
in an unknown indoor environment  
with obstacles and corners**

Thesis submitted in partial fulfillment of the requirements  
for the M.Sc. degree

By  
Dror Katz

Under the supervision of: Prof. Yael Edan

July 2016  
Beer-Sheva



Ben-Gurion University of the Negev  
Faculty of Engineering Sciences  
Department of Industrial Engineering and  
Management

**Development of algorithms for a  
human-following robot equipped  
with Kinect vision and laser sensors  
in an unknown indoor environment  
with obstacles and corners**

Thesis submitted in partial fulfillment of the requirements  
for the M.Sc. degree

Dror Katz

Under the supervision of: Prof. Yael Edan

Author: Dror Katz



Supervisor: Prof. Yael Edan



Chairman of Graduate Studies Committee: Dr. Yisrael Parmet



July 2016



## Acknowledgments

This research would not have been possible without the support of many people.

I wish to express by deepest gratitude to my advisor, **Prof. Yael Edan**. Thanks for the support, guidance, assistance and helpful advices. I learned a lot from you.

I wish to thank CDI (Boaz, Ziv and Avihai), for letting me use their spaces and equipment as much as I needed.

Special thanks to all the wonderful people of the BGU industrial engineering department that were involved during the research, always with a huge smile and help to solve problem during the progress. **Shanee Honig, Moshe Bardea, Shai Givati, Prof. Tal Oron Gilad, Dr. Guillaume Doisy, Polina Kurtser and Inez Mureinik.**

At last and most important, I wish to thank my wife Maayan, my family and friends, for their moral support through the entire process of writing this thesis.

This research was supported by the Ministry of Science, Technology & Space, Israel, Grant # 47897, "Follow me", the Helmsley Charitable Trust through the Agricultural, Biological and Cognitive (ABC) Robotics Center, and the Rabbi W. Gunther Plaut Chair in Manufacturing Engineering, all at Ben-Gurion University of the Negev.

## Abstract

A robot's ability to follow a human in an unmapped indoor environment is fraught with challenges due to unknown obstacles, unexplored walls and unfamiliar corners and corridors. This research aimed to develop human-following robot algorithms that reduce the number of instances of loss of the human and to improve the robot's ability to self-recover in unknown environments. To this end, five algorithms and two human-following methods were developed and tested in a series of experiments with a mobile robot platform.

Algorithms. The developed algorithms do not use any a-priori information about the environment (i.e., operate with no a-priori mapping) and do not require that the human have any particular carry-on item or specific clothing.

The algorithms use depth methods to improve the occlusion detection process. They use a laser sensor to avoid obstacles during the following process in real time, adapt to the linear and angular velocities of the robot and remember the last position of the person to search the person if the person disappears by moving to the last known position of the person and turning in the direction that was calculated before loss.

The following five algorithms were developed: real time **Obstacles-Avoidance** by laser (**OA**), **Search-After-Disappear** to search for the person after tracking has been lost (**SAD**), and three occlusion detection algorithms, namely, **Depth-Occlusion-Detection (DO)** using the depth information from the Kinect sensor (see below), **Vision-Occlusion-Detection (VO)** using the 2D information from the Kinect, and **Combined-Occlusion-Detection (CO)** using both depth information and 2D information.

Sensors. Two sensors were used for detecting humans: 1) Kinect using OpenPTrack (<http://openptrack.org/>) that detects people standing on the ground by using histogram of oriented gradients (HOG ) and support vector machine (SVM) classifiers and 2) the on-board laser SICK300, which can detect the human's legs at 20 cm above the ground.

Methods of following. Two main human-following methods were developed and evaluated. A *Direct-Following (DF)* method, in which the robot moves directly towards the person being detected, and a *History-Following (HF)* method that causes the robot to move to previous positions of the person. The evaluation of the methods comprised two stages: 1) using only the Kinect to follow the human, denoted as non-adaptive methods, and 2) using both sensors to follow the person (if the Kinect loses the person, then the method uses the laser), denoted as adaptive methods.

Robot platform. The algorithms were implemented on a Pioneer LXRobot mobile platform equipped with a Kinect and laser sensor.

Experiments – methodology and results. To overcome the difficulties inherent in an unknown environment, real-time algorithms were developed and integrated in various combinations with the two main human-following methods (*DF*, *HF*). A series of experiments were conducted to derive best fit parameters and to evaluate the algorithms. Performance measures applied for the comparison were the number of instances of loss of the human, number of self-recoveries of the robot and the number of safety interventions, the distance between the robot and the human, the length of the robot path, reliability of the legs detector, reliability of occlusion detections, and the ratio of stable tracking (percent of stable tracking from the entire trial) of the Kinect and laser.

The first two experiments were preliminary experiments to choose the robot following parameters. After implementing the Kinect V2 with a Pan Mechanism, an experiment was conducted to test the objective and subjective metric performances of three following angles (0, 30, 60°). The results indicated that there was no significant difference between 0° and 30° following angles. The results for following at 60° were not sufficiently reliable.

In an experiment aimed to compare the different occlusion detection algorithms, the ***DO*** yielded the best performance. An experiment to determine the best combination of the three algorithms (***DO***, ***OA***, and ***SAD***), once with the *DF* method and once with the *HF* method, indicated that combining the three algorithms yielded best performance. The final, and most important experiment, compared the two methods of human-following (*DF* and *HF*) with the combination of the three algorithms to same chosen methods with addition laser legs detector (denoted as adaptive methods) for use if necessary if the Kinect lost the participant.

Conclusions. The results showed that adaptive methods that combine the Kinect and laser sensor to follow the person were better than non-adaptive methods (the algorithms that use only the Kinect to follow the person) and that *direct-following* methods are better than *history-following* methods.

# Table of Contents

Abbreviations.....	1
<b>1. Chapter One: Introduction .....</b>	<b>2</b>
1.1 Description of the Problem .....	2
1.2 Objectives .....	3
1.3 Thesis contributions .....	3
1.4 Thesis structure .....	4
<b>2. Chapter Two: Literature Review .....</b>	<b>5</b>
2.1 Introduction.....	5
2.2 Applications .....	5
2.3 Tracking Methods .....	5
2.4 Navigation.....	11
2.5 Occlusions and Target Loss .....	12
<b>3. Chapter Three: Methodology .....</b>	<b>13</b>
3.1 General.....	13
3.2 Robot Hardware and Software.....	13
3.3 Algorithms .....	15
3.4 Experimental Design.....	17
3.5 ROS implementation.....	27
3.6 Analysis.....	28
<b>4. Chapter Four: Algorithms .....</b>	<b>31</b>
4.1 Overview.....	31
4.2 Depth Occlusions Detection.....	32
4.3 Vision Occlusions Detection.....	35
4.4 Combination of Depth and Vision Occlusions Detection .....	38
4.5 Obstacle Avoidance .....	38
4.6 Search-after-Disappear.....	39
4.7 Kinect Orientation Control – Pan Mechanism.....	39
4.8 Direct-Following Method .....	42
4.9 History-Following Method .....	44



4.10 Adaptive Following Methods (Kinect and Laser).....	47
<b>5. Chapter Five: Results and Discussion.....</b>	<b>49</b>
5.1 Overview.....	49
5.2 Preliminary Experiment: Identifying Testable Parameters on a No-Pan Kinect V1.....	49
5.3 Preliminary experiment: Testing Objective & Subjective Performances of 2 Angles.....	50
5.4 Testing Objective & Subjective Metric Performances of 3 Angles on a Pan Kinect V2.....	54
5.5 Comparison of Occlusion Algorithms .....	55
5.6 Direct-Following Experiment .....	56
5.7 History Following Experiment.....	56
5.8 Adaptive Kinect-Laser vs. Non-Adaptive (Direct and History-Following) Experiment .....	62
<b>6. Chapter Six: Conclusions and Future Work.....</b>	<b>71</b>
6.1 Conclusions.....	71
6.2 Research limitations.....	71
6.3 Future Work.....	72
<b>7. References.....</b>	<b>73</b>
<b>8. Appendices</b>	
Appendix A- An explanation of how to start the following methods and integrated algorithms.	75
Appendix B- Likert-Style Questions for Section 3.4.3.....	77
Appendix C- Statistical Analysis.....	78
Appendix D- C++ codes.....	106

## List of Figures

Figure 1- Comparison of the specifications of Kinect Version 1 and Kinect Version 2.....	13
Figure 2- Pioneer LXRobot with Kinect.....	14
Figure 3- System hardware and software: computers, devices, sensors and algorithms.....	15
Figure 4- Human-following methods and connections of the algorithms.....	16
Figure 5- Methodology sequence of steps .....	17
Figure 6- Experimental steps .....	18
Figure 7- Walking path .....	20
Figure 8- Addition of the cardboard "trunk" .....	21
Figure 9 - Direct-Following and History-Following steps.....	24
Figure 10- Adaptive vs. Non-Adaptive experiment path .....	26
Figure 11- Nodes and topics flow-chart.....	27
Figure 12- New BBC parameters.....	33
Figure 13 - DO pseudo code .....	34
Figure 14 - DO person detection with a half-BBC (from OpenPTrack) near a right wall (left). Person's Distance of 2.4 m with a small right occlusion and right wall detection without a left occlusion (right) .....	35
Figure 15 - VO pseudo code .....	37
Figure 16 - green VO person contour with red straight line of a wall (left). MONO image with a half hidden person with wall from the right (center). Right wall detection (right) .....	38
Figure 17 - Obstacles corridor examples .....	40
Figure 18 - Laser to world coordinates .....	45
Figure 19 - Kinect to world coordinates .....	46
Figure 20 - Flowchart of the decision making Kinect- and laser-detection algorithm .....	48
Figure 21 - Examples of losing tracking at distances $> 4$ m and at close distances.....	51
Figure 22 - Losses reasons (back-following vs. side-following) .....	51
Figure 23 - Means and variance degrees (back-following vs. side-following).....	52
Figure 24- An example of Direct-Following with search and occlusion algorithms from RVIZ .....	58
Figure 25 - Example of History-Following with search and occlusion algorithms from RVIZ .....	59
Figure 26 - Five trials total path length of the robot .....	61
Figure 27 - Means for groups in homogeneous subsets.....	66
Figure 28- Trials examples for the adaptive and non-adaptive experiment.....	69
Figure 29- Descriptive Direct vs History .....	88
Figure 30- Descriptive Adaptive vs Non-Adaptive .....	92
Figure 31- Descriptive Direct vs History .....	95
Figure 32- Descriptive Gender (Males vs females) .....	96
Figure 33- Descriptive Subjects (24) .....	98

## List of Tables

Table 1- Five combinations of trials .....	23
Table 2- Combinations of methods and algorithms .....	25
Table 3- Statistical analyses .....	30
Table 4- Properties of Kinect vs laser detector .....	48
Table 5- The set of usable parameters tested for the variables .....	49
Table 6 - Following angle results (back-following vs. side-following).....	52
Table 7- Subjective results of the performances of two angles experiment.....	54
Table 8- Cumulative results for objective measures for quality of walk .....	55
Table 9- results of comparison occlusion algorithms experiment .....	56
Table 10 - Results for Direct-Following experiment .....	57
Table 11- Results for History-Following experiment .....	59
Table 12- Statistic comparison between 3 DF trials vs 2 HF trials .....	60
Table 13 - Results for the comparison between 3 DF trials vs 2 HF trials .....	60
Table 14 - Results for five trials.....	61
Table 15 – Statistic comparison between Adaptive Kinect-laser methods vs Non-adaptive Kinect methods.....	62
Table 16- Direct-Following (adaptive and non-adaptive methods) vs History-Following (adaptive and non-adaptive methods) .....	63
Table 17- Statistical comparison between males vs females .....	64
Table 18- Statistical comparison between the four trials .....	65
Table 19- The summarized results of the adaptive and non-adaptive (DF and HF) experiment ranking .....	70
Table 20 - Raw data occlusion's algorithms comparison .....	79
Table 21- ANOVA occlusion's algorithms comparison .....	80
Table 22- Tukey HSD occlusion's algorithms comparison.....	80
Table 23- Total_True occlusion's algorithms comparison .....	81
Table 24- True_Wall occlusion's algorithms comparison.....	81
Table 25- Total_False occlusion's algorithms comparison .....	81
Table 26- PEARSON correlations occlusion's algorithms comparison.....	82
Table 27- Raw data Direct-Following and History-Following .....	83
Table 28- ANOVA Ratio_Track_Kinect Direct-Following .....	83
Table 29- Tukey HSD Ratio_Track_Kinect Direct-Following.....	84
Table 30- Ratio_Track_Kinect Direct-Following.....	84
Table 31- Descriptive Total_Loss Direct-Following .....	84
Table 32- ANOVA Subject's_average_velocity Direct-Following.....	85

Table 33- Tukey HSD Subject's_average_velocity Direct-Following.....	85
Table 34- ANOVA Total_path_distance History-Following.....	86
Table 35- Homogeneity of Variances Direct vs History.....	86
Table 36- ANOVA Direct vs History .....	87
Table 37- Homogeneity of Variances Total_path_distance- 5 trials .....	88
Table 38- ANOVA Total_path_distance- 5 trials .....	89
Table 39- Tukey HSD Total_path_distance- 5 trials .....	89
Table 40- Total_path_distance- 5 trials .....	89
Table 41- Raw data Adaptive vs Non-Adaptive (Direct and History).....	90
Table 42- Homogeneity of Variances Adaptive vs Non-Adaptive .....	91
Table 43- ANOVA Adaptive vs Non-Adaptive.....	91
Table 44- ANOVA Direct vs History .....	93
Table 45- Homogeneity of Variances Gender (Males vs females) .....	95
Table 46- ANOVA Gender (Males vs females).....	95
Table 47- Homogeneity of Variances Subjects (24) .....	96
Table 48- ANOVA Subjects (24).....	96
Table 49- Tukey HSD Subjects (24).....	97
Table 50- Homogeneity of Variances Trials (4) .....	98
Table 51- ANOVA Trials (4).....	99
Table 52- Tukey HSD Trials (4) .....	100
Table 53- Total_Loss Trials (4) .....	101
Table 54- Safety_Intervention Trials (4) .....	102
Table 55- Average_Distance Trials (4).....	102
Table 56- Ratio_Track_Kinect Trials (4).....	103
Table 57- False_Depth_Occlusion Trials (4).....	103
Table 58- Depth_Occlusion Trials (4) .....	104
Table 59- Subject's_Velocity Trials (4) .....	104
Table 60- Robot's_Distance Trials (4) .....	105

**This thesis is in part based on the following publications:**

**Conference Publications:**

Honig, Katz, Oron-Gilad, Edan. 2016. " The Influence of Following Angle on Performance Metrics of a Human-Following Robot" in ROMAN.

**Journal Papers to be submitted:**

Katz, Edan. 2016. "Kinect and laser based algorithms for a human-following robot operating in unknown indoor environments" (in preparation).

# Abbreviations

## General

BBC	bounding box coordinates
CDI	Center for Digital Innovation
DLLM	double-layer locating mechanism
HOG	histogram of oriented gradients
LRF	laser range finder
OBB	oriented bounding box
PCL	point cloud library
ROI	regions of interest
ROS	robot operating system
SDK	software development kit
sig.	level of significance
SLAM	simultaneous localization and mapping
STD	standard deviation
SVM	support vector machine

## Algorithms

<i>CO</i>	Combined-Occlusions-Detection
<i>DO</i>	Depth-Occlusions-Detection
<i>OA</i>	Obstacle Avoidance
<i>SAD</i>	Search-after-Disappear
<i>VO</i>	Vision-Occlusions-Detection

## Methods

<i>DF</i>	Direct-Following
<i>HF</i>	History-Following

# **1. Chapter One: Introduction**

## **1.1 Description of the Problem**

In robotics, algorithms for human-following robots have been the subject of intensive research (Jia et al. 2016; Li et al. 2015; Sahoo and Ari 2015; Jung et al. 2014; Karakaya et al. 2014.; Doisy et al. 2012; Machida et al. 2012; Motai et al. 2012), but one of the main problems, namely, the robot losing track of the person it is following, remains to be resolved (Ota et al. 2013). Additional problems are caused by occlusions, quick turning of the person, and crowded environments or they may occur when the target person is obscured by obstacles. There are many approaches in the literature to overcome these problems (Ota et al. 2013; Granata et al. 2011; Kim et al. 2010; Ma et al. 2008; Kmiotek and Ruichek 2008; Huang et al. 2007), but to date, none is completely satisfactory. One option is to provide the robot with a detailed map of the environment that can be used to avoid obstacles and to predict the next step in a corridor or at a corner. The disadvantages of this option are that requires a-priori information about the environment and that it lacks versatility. For the robot to avoid obstacles without having a-priori information, it must be equipped with a real-time obstacle-detection algorithm so that it can react immediately when an obstacle appears in its close vicinity. Another option uses a decision-making engine that selects from different following methods, such as direct following or path following, when the person disappears (Granata et al. 2011). An approach that uses vision-based face detection (Huang et al. 2007) could be used in the following system to restore tracking after target loss, but the disadvantage of this approach is that the person must face the robot. Yet another method to maintain following is for the person to carry a device for guiding the robot, like a dog-leash (Young et al. 2011), but once again, such a solution reduces the versatility of the robot and has the additional disadvantage that the person has to carry the device.

Most human-following algorithms are programmed to follow a person directly from behind (Granata et al. 2011). However, if the robot 'realizes' that something is blocking its line of sight to the person, it can prevent the interruption in following by changing its tracking angle through increasing its line of sight with the person (wider angle). When a person starts to disappear, due to, say, rounding a corner, the robot must react to reduce the probability of losing the person.

## 1.2 Objectives

This study aimed to develop human-following algorithms for robots that reduce the number of instances of the robot losing the human and that improve the robot's ability to self recover in environments that branch or have unexpected corners, obstacles or occlusions. In this thesis, the focus was directed to detecting corners and obstacles that interrupt the line of sight between the robot and the person by using a Kinect vision sensor and Kinect people recognition, such as Skeleton or OpenPTrack (Munaro et al. 2014). The study explored whether and how the use of vision and depth methods can contribute to improving detection in an occlusion situation. It also explored whether following the human at different angles to create a better line of sight between the robot and the person could reduce the probability of losing track of the person. To enable the robot to deal with obstacles during the following process, a real-time obstacle-detection algorithm was developed. The algorithms do not use any a-priori information about the environment (i.e., it operates without mapping) and do not rely on any special carry-on item or any specific clothing of the human.

## 1.3 Thesis contributions

This thesis introduces four algorithms and two human-following methods, which were developed to overcome the difficulties – due to unexpected obstacles, unexplored walls and unfamiliar corners and corridors – in following a human for robots operating in unmapped environments. Here, a brief summary is provided, with more detailed definitions and explanations being given later in the relevant sections of the thesis.

### *Algorithms*

- Real-time obstacle detection and avoidance without a-priori information about the location of the obstacles or any kind of pre-built map of the environment. The algorithm declares an adaptive corridor in front and on the sides of the robot to narrow the scan area, which depends on the prevailing linear and angular velocities of the robot. As the robot turns, the corridor moves to the side of the turn to enable the robot to search for obstacles inside the turning radius. The angular velocity depends on the side and distance of the obstacle from the center of the robot.
- Real-time occlusion detection using depth information of the pixels. The algorithm compares the depth value of the pixels (distance value of the pixels) inside the bounding box coordinates (BBCs) of a detected person to the distance of the whole person from the robot and searches for small values that indicate closer pixels (indicating an occlusion). It does not use the ground depth value but depends on the person's distance

from the robot. The algorithm reduces false alarms and can detect both small and large occlusions and even a vertical occlusion like a wall.

- Real-time occlusion detection using 2D images by transforming the pixels' coordinates of the depth image to a 2D (MONO) image. The algorithm finds contours of the whole person and any straight vertical lines that indicate an occlusion.
- Search for the person after disappearance. The algorithm moves the robot to the person's last known position and turns the robot in the direction that is calculated according to the last few frames obtained before the person had disappeared.

### ***Following methods***

- Direct human-following method moves the robot directly to the position of the detected person. The method gives priority to sending the robot the linear and angular velocity to avoid any obstacles change. The robot can then change the following angle according to the occlusion-detection algorithm, which can work with robots equipped with Kinect and/or laser sensors.
- History human-following method moves the robot directly to the historical position of the person being tracked. The method avoids big changes of the position of the person caused by the movements of the robot and the Kinect, avoids quick turns that cause the Kinect to lose the person and avoids problems with two sources of person detection (Kinect and laser).

## **1.4 Thesis structure**

The thesis is organized as follows: Chapter 2 presents a review of the literature on human-following methods by several sensors, robot navigation, detection of occlusions, and target loss. Chapter 3 presents the methodology, which starts with a description of the robot hardware and software and continues with a description of the sequence of experimental steps and all the experimental procedures. This chapter ends with a description of the implementation of the robot operating system (ROS) and analysis procedures. Chapter 4 presents the description of the algorithms. Chapter 5 gives the results of the experiments and some discussion of the results. Chapter 6 presents conclusions and recommendations for future work.



## **2. Chapter Two: Literature Review**

### **2.1 Introduction**

Tracking of a person by a robot has been intensively investigated in recent years (Jia et al. 2016; Sahoo and Ari 2015; Li et al. 2015; Karakaya et al. 2014.; Jung et al. 2014; Doisy et al. 2012; Machida et al. 2012; Motai et al. 2012), with the main thrust of the research being devoted to three challenging tasks related to person-tracking robots: (1) robot navigation, (2) tracking methods, and (3) problems associated with occlusions and target loss. Recently, research has focused on new tracking methods and on the fusion of several methods to investigate problems related to recovery after target loss or occlusions (Ota et al. 2013; Granata et al. 2011; Kmiolek and Ruichek 2008; Huang et al. 2007).

### **2.2 Applications**

Robot tracking has been applied for many uses and applications, including, among many others: (1) Care robots in nursing, such as those providing medical help for the elderly, where the robot helps to carry medicines and treat the patients (Machida et al. 2012); (2) Robots for assisting workers to assemble large equipment (Karakaya et al. 2014); (3) Robots that help passengers to carry heavy luggage at airports and train stations (Li et al. 2013); (4) Smart shopping carts, such as the Kinect grocery cart, that follow the customer and scan his/her purchases in the supermarket or the mall; (5) Robots as walking assistants to support a person (Jia et al. 2016); (6) Museum guidance robots that follow a person and provide guidance and information (Karakaya et al. 2014); and (7) Robots designed to help the disabled (Jia et al. 2016).

### **2.3 Tracking Methods**

#### **2.3.1 Vision-Based Tracking**

Along the years, many algorithms have been developed for use with different sensors and methods that rely on smart environments (Najmaei and Kermani 2011). Most person detection and tracking methods use vision-based techniques (Jia et al. 2016; Sahoo and Ari 2015; Y. Li et al. 2015; Yao et al. 2012; Motai et al. 2012; Ma et al. 2008), but such techniques have some inherent disadvantages, such as sensitivity to illumination changes (Liu et al. 2004) or problems with computing size or large processing. Two examples of vision-based algorithms are the particle filtering (PF) algorithm and the mean-shift algorithm

(Yao et al. 2012). Particle filters represent a distribution by a set of weighted samples called particles. Each particle is a guess representing one possible location of the object being tracked. This weighted distribution is updated along time by using a set of equations. The mean-shift algorithm creates a confidence map in a new image based on the color histogram of the object in the previous image, and applies mean-shift optimization to find the peak in a confidence map near the object's previous position. Each pixel of the new image is a probability, which is related to the probability of the pixel color occurring in the object in the previous image. An advanced mean-shift algorithm can be used for detection of a person by modeling a star skeleton of the human body and adding a combination of a block search algorithm for high-speed movement and a target loss recovery algorithm (Sahoo and Ari 2015). Some vision-based methods are based on detection of the person's legs (Li et al. 2015); for example, they divide the tracking into global tracking, which describes the motion of both feet as one element, and local tracking, which describes the relation between the two feet (Li et al. 2015).

### 2.3.2 Laser-Based Tracking

Rather than using vision-based techniques, several methods use laser sensors to 'find' the human (Karakaya et al. 2014; Jung et al. 2014; Alvarez-Santos et al. 2012; Kim et al. 2010; Sales et al. 2010; Ma et al. 2008), since the laser range finder (LRF) can provide more precise position information. A laser-based technique has the additional advantages that it requires fewer processing sources and is less influenced by lighting conditions (Kim et al. 2010). In addition, the laser is not influenced by colors or face detection. The main disadvantage of laser-based techniques lie in their ability to recognize only items in the line of sight, which is relatively a small area when compared to that covered by vision-based methods.

### 2.3.3 Integration of Sensor-Based Tracking Techniques

Several studies have described methods of integration of vision-based sensors and lasers, for example, such methods may use vision-based methods to extract the body and lasers to measure distance (Ma et al. 2008). In most applications, the LRF detects the legs of a person (Alvarez-Santos et al. 2012), but human legs can be confused with chair or table legs, so there is a need to compose a map of the environment by using a sensor fusion technique (Motai et al. 2012). In such a scenario, the laser can assist the robot to navigate and to avoid obstacles and not only to track the person.

A combination of depth images and thermal images has also been developed for detecting more than one person (Hadi et al. 2015). A depth image is fused with the region of interest (ROI) obtained from the thermal image to derive a person's contour. Occlusions of the detected persons are resolved using BBCs. The algorithm has two stages: The first is a pre-detection stage to obtain the BBC from the thermal image representing the region of humans. In the second stage, people are detected with contours of depth measurements, and an occlusions detector classifier is applied to detect people that are occluded.

#### 2.3.4 Depth-Camera-Based Tracking

Yet another tracking method focuses on the use of time-of-flight (TOF) range cameras (Ikemura and Fujiyoshi 2011; Plagemann 2010; Plagemann and Koller 2010). Many algorithms have been proposed to address the problem of pose estimation and motion capture from range images, for example, a filtering algorithm to track human poses using a stream of depth images captured by a TOF camera (Plagemann 2010). There have been several works on detection of human parts using TOF cameras (Plagemann and Koller 2010). Examples include: (i) using a point detector to solve problems of detection and to identify body parts in depth images (Plagemann and Koller 2010) and (ii) using a window-based human detection method by comparing depth similarity features based on depth information (Ikemura and Fujiyoshi 2011).

When there is a problem with low visibility conditions, such as in smoky environments, the vision-based sensors or lasers do not provide good solutions (Sales et al. 2010). In such a case, the use of LRF and sonar sensors is proposed in combination with a vision-based system that can determine the amount of smoke in the environment and then decide on the optimal combination of sensors for the particular conditions (Sales et al. 2010).

#### 2.3.5 Kinect-Based Tracking

In June 2011, Microsoft released the Kinect software development kit (SDK). This SDK allows developers to write Kinect apps for the Kinect sensor. Kinect is an RGB-D sensor that provides depth images, allowing real-time object segmentation, which based on a distance gradient. The depth sensor, which includes an infrared laser projector in combination with a monochrome sensor, captures video data in 3D under any lighting conditions, and hence facilitates the development of more efficient human-tracking robots (Pucci et al. 2013). Indeed, Kinect may be regarded as a breakthrough in the field: it has

made possible new approaches and techniques for person-tracking research (Machida et al. 2012; Doisy et al. 2012; Ikemura and Fujiyoshi 2011). The 3D position information from the Kinect sensor enables the velocity and attitude of the mobile robot to be controlled directly. A Kalman filter can be used to reduce the noise and to estimate the human's motion (Machida et al. 2012). Another Kinect detection example, which detects humans using a 2-D head contour model and a 3-D head surface model, was developed (Ikemura and Fujiyoshi 2011).

OpenPTrack is an open source algorithm based on the Robot-Operation-System "ROS" (Quigley et al. 2009) and Point Cloud Library (PCL) (Rusu and Cousins 2011). It detects people with the Kinect sensor using a histogram of oriented gradients (HOG) and a support vector machine (SVM) learning classifier for creating the confidence, based on a large training dataset, that the detected area is a person. In addition, an unscented Kalman filter (for nonlinear systems) is exploited to predict people's positions and velocities along the two ground plane axes (X,Y) (Munaro and Menegatti 2014). The algorithm provides many person-specific parameters, namely: the position of the person in the world, as 'seen' by Kinect, the coordinates of the bounding box around the person in the depth image, height of the person, distance to the person and more. The algorithm is able to track people at 30 Hz with minimum latency on the assumption that the plane on which people stand or walk is the ground, and therefore the number of Regions-Of-Interest "ROIs" that are candidates to contain people is reduced. After selecting a set of clusters from the point cloud, the algorithm processes a Histogram-of-Oriented-Gradients "HOG"-based people detector applied to the corresponding image patches (Munaro and Menegatti 2014), and finally it uses a SVM classifier for deciding on the confidence that the patch is a person. The main advantage of this algorithm is its ability to use both RGB and depth information for obtaining the best result when the RGB image is good, while using depth data alone if the RGB image is too dark.

### 2.3.6 Different Detectors for Different Distances

For close-range detection (up to 5-7 m), the real-time RGB-D based multiperson detection and tracking system of Jafari, Mitzel, and Leibe (2014) uses an extremely fast depth-based upper-body detector and for further distances it adds an appearance-based full-body HOG detector. The idea is to use the depth information for ROI extraction to detect people at close range, where depth measurements are reliable, while simultaneously extrapolating scene geometry information to constrain the search space for appearance-based people detection

in the far range. The system uses a template of an upper body to detect people by using depth information at close range; the system computes a distance matrix consisting of the Euclidean distances between the template and each normalized depth image segment. For detecting people at further distances, the system uses HOG alone on the ROIs, according to the ground plane. Finally, the detections are converted to ground plane coordinates and are associated into trajectories using an extended Kalman filter.

### 2.3.7 Outdoor Tracking

Most tracking systems or mobile robots have been designed to operate in indoor environments (Karakaya et al. 2014). Robotic systems designed to operate in outdoor environments, including person tracking and avoiding moving obstacles, in a crowded environment are very rare, because of the noise generated by the outdoor environment. Nonetheless, a robot capable of following a marathon runner has been developed; it uses a laser to detect the runner and to avoid obstacles (Jung et al. 2014). This robot, known as MSR (Marathoner Service Robot), is designed to carry water and equipment for the runner.

## 2.3.8 Examples of human tracking

Examples of human-tracking algorithms for robots are given in the table below.

Sensor	Algorithm	Experiment	Advantages	Disadvantages	Indoor/ outdoor	Variable speed	Object	Quick turn	Obstacles	Crowd/ single	Review
Kinect	Kalman filter	Circular path vs zigzag	Low price; depth help for clutter background	Velocity less than 1 m/s. Light condition, Spatial resolution, Sample rate.	Indoors	Yes	Bone model (all body)	No	no	single	Machida et al. 2012
Kinect	A 2D edge detector and a 3D shape detector to utilize both the edge information and the relational depth change information in the depth image.	Two persons indoors with many objects in the vicinity	Easily adjust to new datasets; no training needed; the first layer reduces computational cost; does not assume person's pose for accurate detection.	High dependency on accurate head detection.	Indoors	Only tracking	Head and then the body	No	only tracking	Up to 2 persons	Ikemura and Fujiyoshi 2011
RGB	Advanced mean-shift.			Does not work for fast motion, prolonged occlusion or changing illumination.	Outdoors	Only tracking	Star skeleton (all body)	No	Only tracking	Single	Sahoo and Ari 2015
RGB	Particle filtering; global tracking (motion of both feet), and local tracking (relative motion of the two individual feet).	Several human walking videos; evaluated against particle filtering.		Fails in noisy backgrounds; shift to another person.	Indoors and outdoors	No	Two feet	No	No	Single	Li et al. 2015
LRF (laser range finder)	Support vector data description (SVDD); human detection algorithm and an avoidance algorithm.	Tracking speed and performance; static and moving obstacles avoidance.		Irregular terrain in the outdoor environment or significant noise from the outdoor environment.	Outdoors	Yes	Torso	Yes	Yes	Crowd	Jung et al. 2014
RGB and LRF	Horizontal Projecting Probability Histogram (HPPH) of upper body with unscented particle filter (UPF) with laser for legs.	Difference distance; another person with the same clothes.	Compares the vision to the distance from the laser.	Another person with same clothes.	Indoors	Yes	Torso for vision, legs for laser.	Yes	No	Crowd	Ma et al. 2008
Infra-red and laser	Kalman filter and optical flow.	Single person indoors, compared to only Kalman-Filter and only Optical-Flow.	Combination of Kalman-Filter and Optical-Flow for quick turn	Crowd.	Indoors	No	Whole body	Yes-sharp turns- OF other KF	Yes (with laser)	Single	Motai et al. 2012
RGB, laser, Sonar	Vision image to determine the degree of visibility of the environment.	Only sonar, only laser, sonar TODA (time difference of arrival)	Low visibility environment.	Laser is better than sonar.	Indoors	No	Legs	No	Yes	Single	Sales et al. 2010

## **2.4 Navigation**

### **2.4.1 Overview**

To navigate in an environment, the human-tracking robot needs to know where all the obstacles are in that environment or to have a pre-built map of the environment that includes the obstacles. In addition, to navigate, the robot must be programmed with a method for following the person.

### **2.4.2 Mapping the Environment**

Most person-tracking approaches in indoor environments are based on wireless networking, such as ultrasound and radio frequency (Garcia-Valverde et al. 2013). For indoor applications, the map of the environment allows safer and more efficient robot navigation, but often the robot must also take into consideration the movement of objects and other people. A study that compared robot navigation with and without a map of the environment showed how the map improved the robot's adaption to the distribution of obstacles (Doisy et al. 2012). A common method for navigation and building a map of the environment is simultaneous localization and mapping (SLAM), one of the most active research and development areas in mobile robotics (Schmidt et al. 2016). Statistical techniques are used to solve SLAM, with the most popular approximate solution being particle filter and extended Kalman filter (Norhidayah and Norida 2015). The main disadvantage of SLAM lies in its computational complexity, which increases significantly with the growing number of landmarks in the environment under exploration (Ding et al. 2015).

### **2.4.3 Obstacle Detection**

The many different sensors employed for obstacle detection include sonar pairs, infra-red measurement sensors, point-to-point laser sensors, LIDAR (light detection and ranging or laser imaging detection and ranging), and Kinect (İ et al. 2012). One such system that combines LIDAR and Kinect uses the LIDAR for obstacle and heading direction and Kinect for eliminating depth data for the immediate environment (Karakaya et al. 2014.; İ et al. 2012).

#### 2.4.4 Methods of Following

There are many methods for navigation and for tracking a person. The robot can track by a direction-following method, in which the robot is always pointing towards the person and moves directly to him. A robot can also track by path-following method, which tracks by adhering to the same path that the person walks. Another – albeit rarely applied – method is parallel-following, which depends on the prevailing state of the environment and on the position of the robot relative to the person (Morales et al. 2012). There are also hybrid methods that use combinations of the above methods (Granata et al. 2011).

### 2.5 Occlusions and Target Loss

A person-tracking robot requires tracking abilities that distinguish between people and objects (Jia et al. 2016; Kim et al. 2010; Ma et al. 2008). Such a robot is also required to overcome problems caused by occlusion, quick turning, and crowded environments, or those that occur when the target person is obscured by obstacles. The robot must also have the ability to recognize loss of tracking due to individuals walking between the mobile robot and the target person and to recover the tracking by using a legs detector (Kim et al. 2010). The development of an adaptive multi-feature mean-shift algorithm in a cluttered environment has been described by Jia et al. (2016) using the double-layer locating mechanism (DLLM). This mechanism takes the course location processing and fine location processing into consideration and is designed to estimate the position of the person by using a combination of data and an ID tag on the person, which can be detected by radio frequency antennas (Jia et al. 2016). For adapting to different moving targets using key characteristics, the robot constructs the target model at the beginning of the tracking process, then detects the human candidates in the scene and finds the target person by using multiple image cues, namely, color and edges (Ma et al. 2008). Other methods to solve the problems of occlusions and target loss were developed. Examples include: (i) Use a decision-making engine when the person disappeared by choosing a different method to follow the person, direct following or path following (Granata et al. 2011). (ii) Continuing to follow even when the sensor lost the target because of a corner (Ota et al. 2013). (iii) Using oriented bounding box (OBB) representation for object tracking (Kmiotek and Ruichek 2008). An approach that uses a vision-based technique for face detection (Huang et al. 2007) could be applied in a following system for successful reinitialization after target loss, but the person must be facing the robot. All of the above examples have been suggested and applied for addressing the problems of occlusions and target loss in the real world (vs in a sterile experimental environment).



### 3. Chapter Three: Methodology

#### 3.1 General

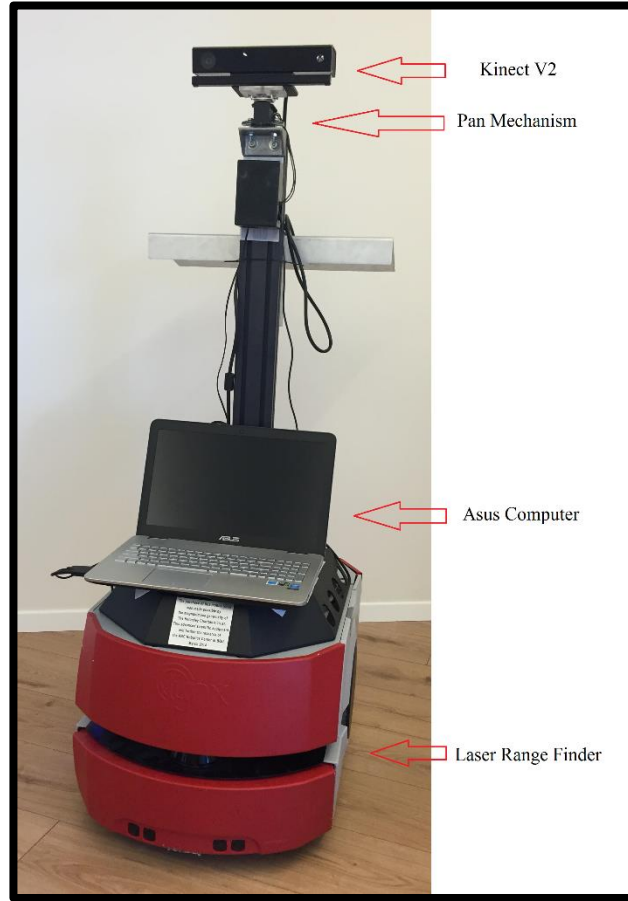
Two main implementations of robot following (direct and history) were improved using five integrated algorithms that were developed in this research. All the developments were created in the ROS in C++. A series of seven experiments implemented on a Pioneer LX Robot were conducted to set up the parameters for the different algorithms and evaluate the algorithms' performances.

#### 3.2 Robot Hardware and Software

All experiments were conducted with a Pioneer LX Robot equipped with front and rear sonic sensors, a SICK S300 laser scanner, a forward bumper panel and a RGB-D camera. In the first and second preliminary experiments, the Pioneer LX Robot was outfitted with a Microsoft Kinect V1 sensor that detects human Skeletons in order to assess the location and distance of the person from the robot (limited to a distance of 4 meters). All the other experiments used a Kinect V2 sensor, which had a better field of view, reaches a distance of 10 meters and improved resolution (**Figure 1**). The Kinect V2 color RGB stream has a resolution of 1920×1080, a horizontal field of view of 84.1° and a vertical field of view of 53.8°. The depth (D) stream has resolution of 512×424, a depth range of 0.4 to 4.5 m, and a horizontal field of view of 70.6°. To facilitate human detection for a wide range of angles, the Kinect V2 was mounted on a Pan mechanism connected by an aluminum rod to the Pioneer LX robot (**Figure 2**).

	Version 1	Version 2
Depth range	0.4m → 4.0m	0.4m → 4.5m
Color stream	640×480	1920×1080
Depth stream	320×240	512×424
Infrared stream	None	512×424
Audio stream	4-mic array 16 kHz	4-mic array 48 kHz
USB	2.0	3.0
Hand Traking	External tools	Yes
Face Traking	Yes	Yes + Expressions
FOV	57° H 43° V	70° H 60° V
Tilt	Motorized	Manual

*Figure 1- Comparison of the specifications of Kinect Version 1 and Kinect Version 2*



*Figure 2- Pioneer LXRobot with Kinect*

To run the implementations of robot following and the integrated algorithms, the system uses two computers (both Asus laptops with Intel core i7-4710HQ processors) in addition to the robot's integrated on-board computer (Intel D252 with 64-bit Dual Core 1.8 GHz) (**Figure 3**). The first laptop is connected directly to the Kinect and the Pan Mechanism. It is responsible for running the OpenPTrack (Munaro et al. 2014) person-detection and *Depth-Occlusions-Detection* algorithms (see Section 3.3) and for controlling the rotation of the Pan. The second computer is responsible for running the main person-following methods (*Direct-Following*, *History-Following*; see Section 3.3), for operating the laser legs detector, for detecting obstacles in real time (*Obstacles-Avoidance*; see Section 3.3) and for the *Search-After-Disappear* algorithm. This second computer also records the ROS information (values of the parameters that are calculated by the running algorithms) and the Rviz information (position of the robot and laser detection) by recording the screen during each trial. Commands are sent to the robot's onboard computer by a TP-LINK router with a wireless speed up to 300 Mbps. An explanation of how to start the methods of following with the integrated algorithms is described in Appendix A. The robot uses a SICK S300 scanning LRF, mounted approximately 20 cm above the ground to detect obstacles and human legs. The ultrasonic sensors and the bumper were not used in this project.

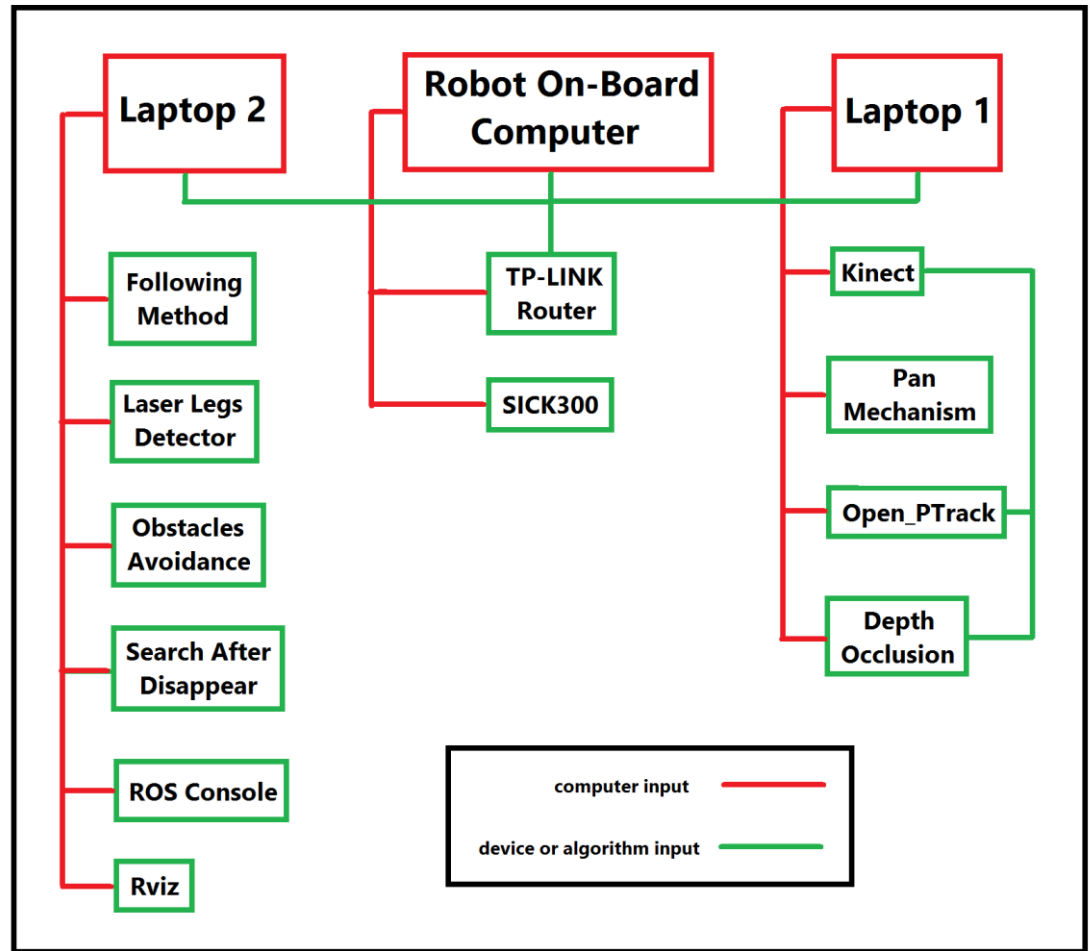


Figure 3- System hardware and software: computers, devices, sensors and algorithms

### 3.3 Algorithms

The following algorithms were developed for obstacle avoidance, search after disappear and occlusion detection (see Chapter 4 for details):

- **Obstacles-Avoidance (OA)** – For the robot to have the ability to detect and avoid obstacles without a map of the environment or any previous knowledge of where the obstacles are located relative to the robot, a real-time obstacle avoidance algorithm was developed. This algorithm scans the environment with a laser at 10 Hz and searches for obstacles in real time. To narrow the search in front and on the sides of the robot, a corridor is declared. The robot reacts only to an obstacle identified within that corridor.
- **Search-after-Disappear (SAD)** – To search for the person, the algorithm remembers the last distance of the person from the robot and subtracts between two values of the horizontal position of the person (the last one and four frames before) to define the direction of the robot's turn. The robot moves to the last position of the person and then turns in the direction determined by the algorithm.

Three different occlusion detection algorithms were developed:

- **Depth-Occlusions-Detection (DO)** – The algorithm compares the depth value of pixels inside the BBCs of a detected person to the distance of the robot from the whole person and searches for small values, which indicate closer pixels. The number of pixels that are closer than the distance of the person is counted by using a threshold that reduces small distance measurement errors and avoids other body parts detected as closer pixels.
- **Vision-Occlusions-Detection (VO)** – The algorithm uses the ROI of a detected person and fuses it with a MONO image (gray-scale) from the Kinect to detect occlusions in a 2D image. After basic image processing, the MONO image searches for straight vertical lines to detect a wall occlusion.
- **Combined-Occlusions-Detection (CO)** – This algorithm uses the **DO** algorithm when the person is close to the Kinect (at distances <5 m) and the **VO** algorithm when the person is far from the Kinect (at distances >5).

Two main following methods were improved, implemented and compared for the human-following robot:

- **Direct-Following (DF)** – The method aims to synchronize all the data from the integrated algorithms, the laser leg detector and the Kinect. This method transforms and calculates the position of the person detected by the Kinect and by the laser, sends avoidance commands to the robot according to the **OA**, changes the following angle according to the **DO**, and implements the **SAD** algorithm.
- **History-Following (HF)** - A semi path follower that moves the robot directly to the historical position of the person was developed. Like the **DF**, this method also synchronizes the data from the integrated algorithms, the laser leg detector and the Kinect detection.

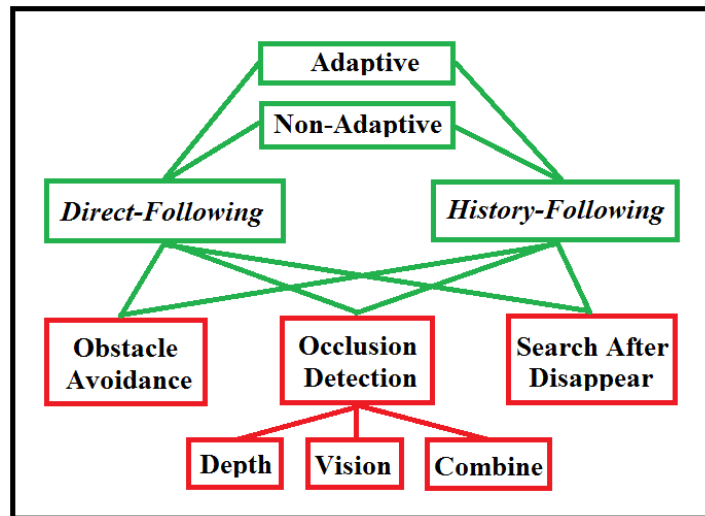


Figure 4- Human-following methods and connections of the algorithms

### 3.4 Experimental Design

#### 3.4.1 Description of the Steps

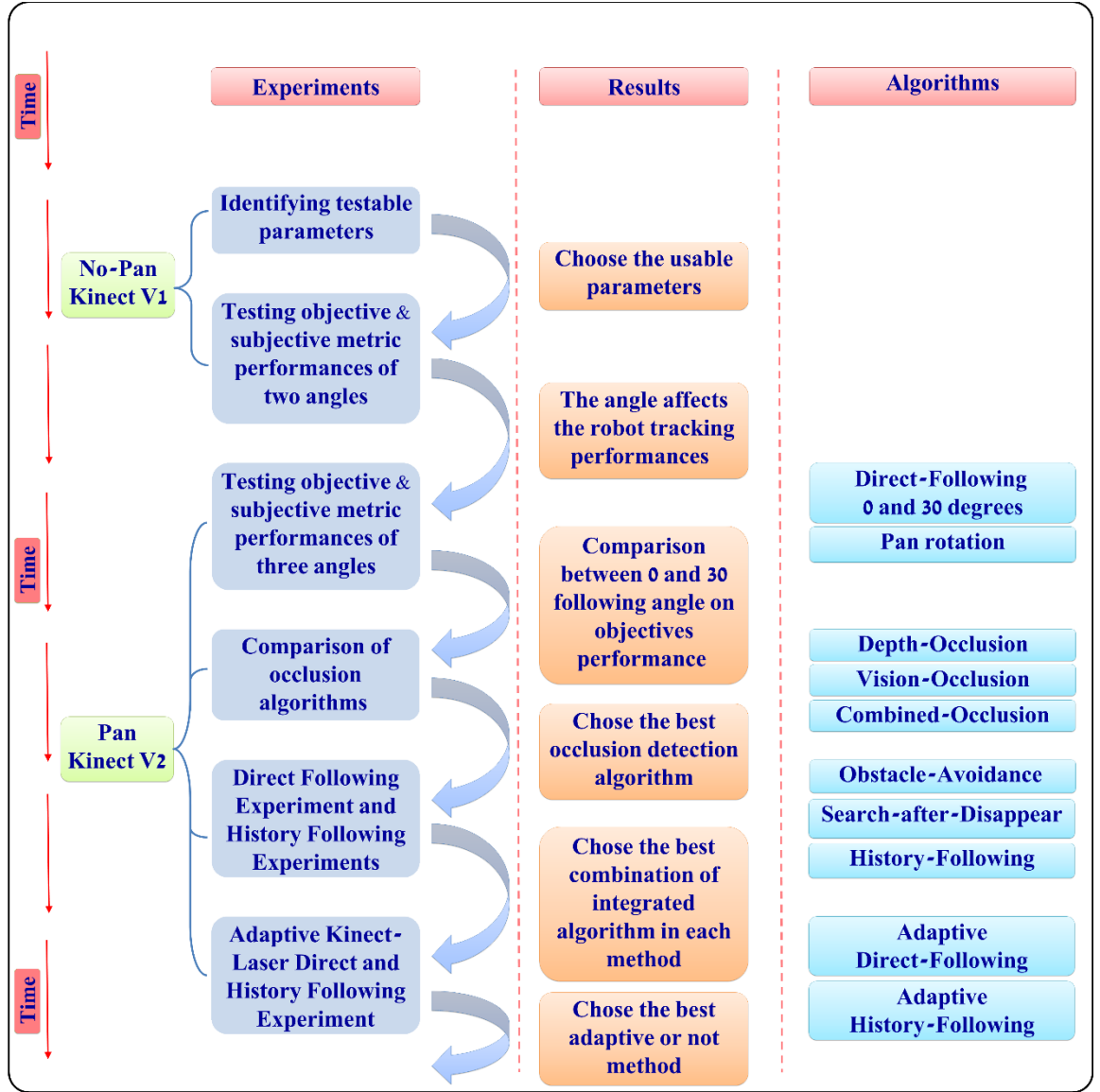
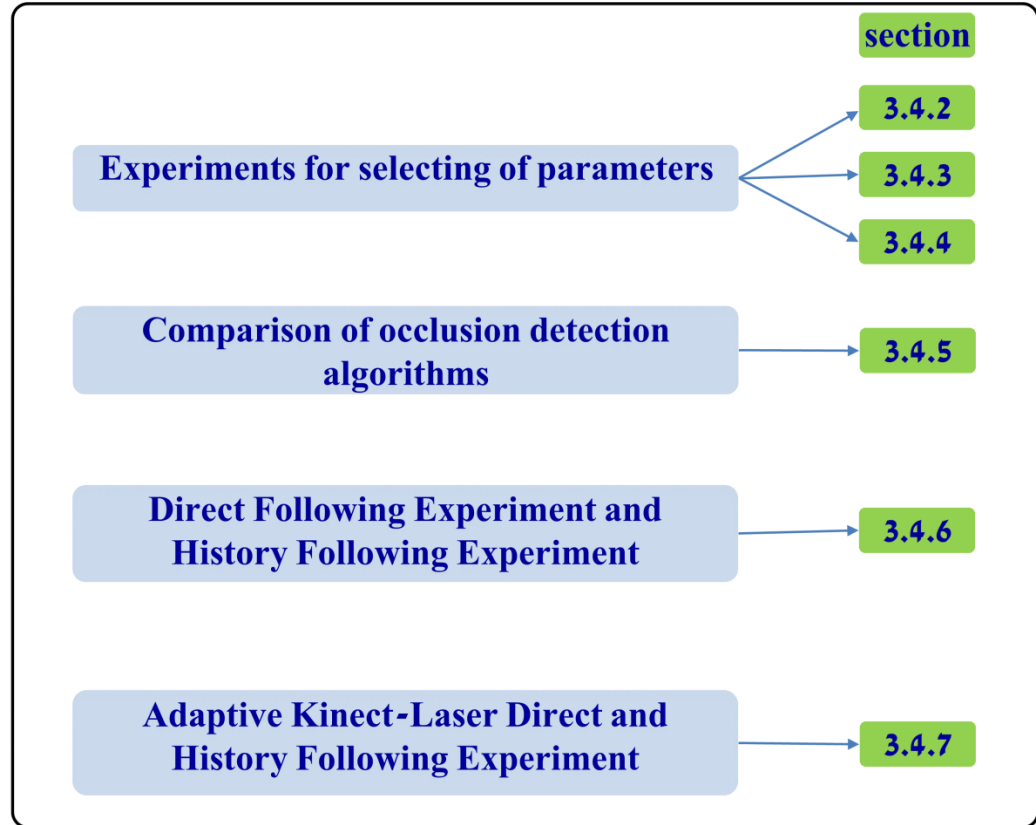


Figure 5- Methodology sequence of steps

The first two experiments were preliminary experiments that used the Kinect V1 Skeleton detection algorithm to choose and test the robot following parameters (see Sections 3.4.2 and 3.4.3). These experiments were completed without a Pan mechanism to rotate the Kinect. From the third experiment onwards, the Pan mechanism was used. After implementing the Kinect V2 with a Pan Mechanism, an experiment was conducted with 24 participants to test objective and subjective metric performances for three different following angles (0, 30, 60°). Occlusion detection algorithms were developed using depth and vision information from the Kinect. The algorithms were compared to identify the one that performs best. Another two integrated algorithms were then developed: one that detects obstacles by laser in real time and the other that searches for a person whenever tracking is lost. These three integrated algorithms (occlusions, obstacles, searching) were then tested once with the *DF*

method and once with the *HF* method to find the best combination in each method. The final and most important experiment compared the two methods of human following (*DF* and *HF*) with the best combination of integrated algorithms (from the results of the experiment described in Section 3.4.6 "*Direct-Following* experiment and *History-Following* experiment") to same chosen methods with the addition of a laser legs detector (denoted as adaptive methods) for use if necessary when the Kinect loses the participant.



*Figure 6- Experimental steps*

### 3.4.2 Preliminary Experiment: Identifying Testable Parameters on Kinect V1 without Pan-Tilt

The aim of this preliminary experiment was select the operational parameters for the following variables:

- Maximum robot speed (m/s)
- Robot responsiveness while walking forward
- Robot responsiveness while turning
- Minimum distance of the robot from the target
- Angle of following.

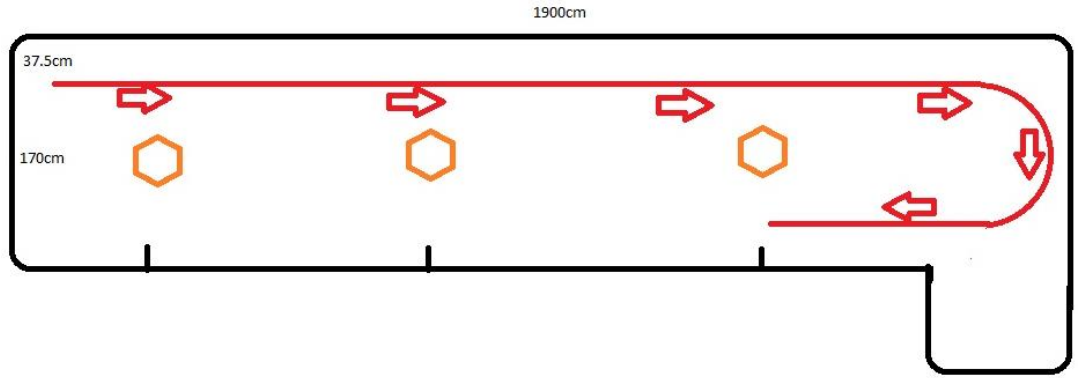
Since a Pan mechanism had not yet been installed and the Kinect was locked in place, the following angle was selected manually. The Kinect V1 has a horizontal field of view of  $57^\circ$  (0.49 rad in each direction).

The following considerations were taken into account when choosing the parameters:

- Selecting an angle parameter that is noticeably different from back-following
- Selecting parameters that lead to the smoothest following possible, that is, with the least number of instances of losing the person and the smoothest robot movement
- Selecting the minimum distance of the robot from the target that is comfortable and comparable to the robot's distance from the 0.49 rad following (not so close that it interferes with personal space and tracking, but not so far that the person does not feel the difference).

### 3.4.3 Preliminary Experiment: Testing Objective & Subjective Metric Performances of Two Angles on a No-Pan Kinect V1

The aim of this experiment was to determine whether the angle at which the robot follows a person affects the human experience and the robot tracking performance. The Pioneer LX was outfitted with a Microsoft Kinect V1 (without a Pan) that detects human Skeletons with the aim to assess the location and distance from the robot of the person. Six subjects (3 female, 3 male) completed a predetermined 25-m track under two conditions: (1) the robot followed directly behind the person ( $0^\circ$  angle), denoted as back-following (2) the robot followed at a  $17.19^\circ$  angle (0.3 rad), denoted as side-following. In order to simulate a real world walking experience, which is rarely constant, linear or without distractions, the walking track included a stop and a turn (**Figure 7**), and subjects were asked to a play game on a smartphone as they walked. The order of the trials was alternated: 3 subjects started with back-following and 3 subjects started with side-following. After each trial (back-following and side-following), subjects were given a questionnaire to assess their experience. In addition, at the end of the study, subjects answered a questionnaire comparing the two conditions. Both surveys were based on Likert-style questions (Appendix B), where the subject had to state how strongly s/he agreed/disagreed with a statement. Subjective and objective performance measures were collected.



*Figure 7- Walking path- preliminary experiment*

To facilitate person following at various angles, a pre-existing human following algorithm was adjusted as follows:

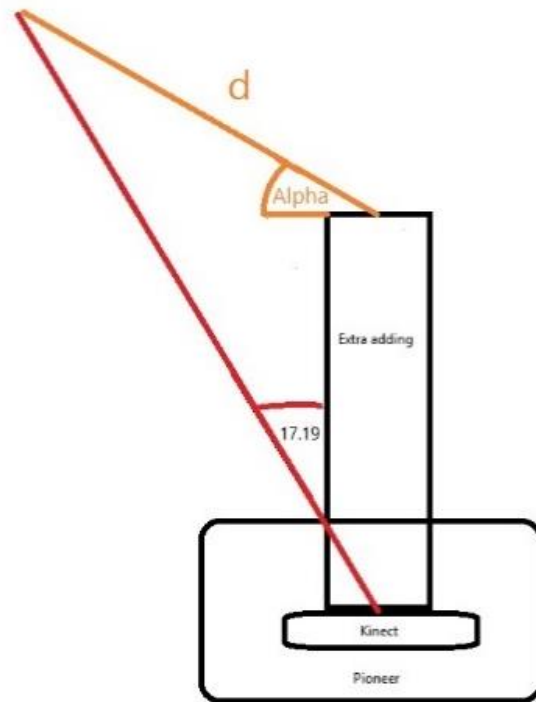
- In the “**side-following**” condition, the "turn acceleration" was set variable to “ $\arctan(y/x)-0.3$ ”, which returns a “0” value at 0.3 rad from the left and stops the robot turning at that point (influenced by the 0.8 responsiveness).
- In the “**back-following**” condition, the "turn acceleration" was set variable to “ $\arctan(y/x)$ ”, which returns a “0” value at 0 rad (at the center of the Kinect camera) in order to ensure that the robot follows directly from behind.

All the other procedures were the same in both following conditions. To make the side-following more efficient, an extra cardboard part was added to the robot to create a wider angle of side-following. The addition of the cardboard “trunk” increased the following angle from the edge of the robot to the person to a 30° angle (**Figure 8**).

The follow parameters were set:

- The Kinect distance was 1 m;
- The responsiveness measure was 0.8 both for turning and walking;
- Maximum robot speed: 0.3 m/s (was slow because the Skeleton image without the Pan kept disappearing from the Kinect's field of view);





*Figure 8- Addition of the cardboard "trunk"*

#### 3.4.4 Testing Objective & Subjective Metric Performances of Three Angles on a Pan Kinect V2

The aim of the experiment was to determine objective and subjective measures to evaluate the quality of following and the perceptions of the subjects toward the robot for three following angles ( $0^\circ$  angle,  $30^\circ$  angle, and  $60^\circ$  angle) under two conditions: when the robot was carrying a valuable personal item (the participant's wallet) or not (Honig et al. 2016). The two conditions were compared on the assumption that increased personal relevance leads to an increase in the involvement felt by the person. This experiment used a mixed between and within-subject design. The wallet manipulation was the between subject variable: 12 participants were asked to place their wallets on the robot for the duration of the study and 13 participants were not. The following angle was the within-subject variable: each participant completed a straight predetermined 20-m walking path under three conditions while being followed by the robot: (1) the robot followed directly from behind ( $0^\circ$  angle), (2) the robot followed at a  $30^\circ$  angle from the left, and (3) the robot followed at a  $60^\circ$  angle from the right. The order of the following angle was counterbalanced between participants. In order to simulate a real-world walking experience, which is rarely constant or without distractions, the walking track included two stops and participants were asked to play a game on a smartphone as they walked. Participants were instructed to walk at their natural walking pace and to stop at two predetermined locations and wait until the robot made a complete stop behind them. The following objective performance measures were selected: distance

and following angle between the robot and the subject, number of instances of loss of the person, and number of interventions. Interventions were classified into two types: interventions due to safety and interventions due to loss. Interventions due to safety were interventions resulting from the robot getting too close to an obstacle or a wall. Interventions due to loss were interventions that were a result of the robot losing track of the person and were made in order to steer the robot back toward the participant.

### 3.4.5 Comparison of Occlusion Algorithms

The aim of the experiment was to compare the occlusion detection algorithms. **DO** uses the depth stream of the Kinect, **VO** uses the MONO stream (gray scale) from the Kinect, and **CO** combines the two algorithms (**DO** below 5 meter, **VO** above 5 meter).

The three algorithms – **DO**, **VO**, and **CO** – were tested and compared with six different distances of the robot to the person and six different person/wall occlusion distances from the Kinect (all in cm): 200/100; 350/200; 500/300; 600/400; 800/600; 400/300 (an occlusion other than a wall). Each distance was tested once for each algorithm.

The **DO** can identify the size of the occlusion (large/small), the direction of the occlusion (left/right) and whether the occlusion is caused by a wall or not. The **VO** can identify only left or right straight vertical lines. When the person is partially hidden and stands without moving, the number of times the algorithm detects the right or left occlusions (large, small or combination of them for the **DO** algorithm) and the number of times the algorithm detects the wall be measured.

### 3.4.6 Direct-Following and History-Following Experiments

The aim of the experiments was to evaluate the performance of various combinations of integrated algorithms in the two main human-following methods and to compare the results.

The two main methods of human-following (**DF** and **HF**) were developed and compared. Each method was tested with various combinations of the three integrated algorithms, **OA** (real time obstacle avoidance by laser), **DO** (depth-occlusion detection using the depth information from the Kinect) and **SAD** (search-after-disappear to search for the person after losing tracking).

After preliminary testing, the maximum linear velocity of the robot was selected as 0.3 m/s for all trials, ensuring sufficient time for the robot to compute and react. The maximum angular velocity during following (not when detecting an obstacle) was chosen as  $0.2\pi$  rad/s.

The experiment was conducted in the offices of the Center for Digital Innovation (CDI) in Beer-Sheva, Israel. The conditions of the experiment included a path of 18-m length with three stops (**Figure 9**). At the beginning of each trial, subjects were asked to stand in front of the robot to allow the robot to detect them. Once the robot had detected the subject, the subject was asked to walk slowly to point 1 (path marks in green), which was marked on the floor, and waits until the robot reaches the third of the six obstacles that had been placed in a line and connected to a demo wall to create a corner (**Figure 9**). When the subject moved from point 1 to point 2, s/he disappeared from the robot's line of sight, simulating how a person would disappear if s/he turned a corner in a hallway. The subject waits at point 2, also marked on the floor, until the robot began to move toward her/him. Once the robot had begun to move toward the subject, the subject moved slowly around three more obstacles and stopped at the last corner 3 (the last obstacle) to wait for the robot. Each subject completed this path five times with different combinations of algorithms (**Table 1**), order counterbalanced (three *DF* and two *HF*). Seven participants took part in this experiment.

*Table 1- Five combinations of trials*

<b>Trial</b>	<b>Following method</b>	<b>Obstacle avoidance</b>	<b>Depth occlusion</b>	<b>Search after disappear</b>
<b>1</b>	History	√	X	√
<b>2</b>	History	√	√	√
<b>3</b>	Direct	√	X	√
<b>4</b>	Direct	√	√	√
<b>5</b>	Direct	√	√	X

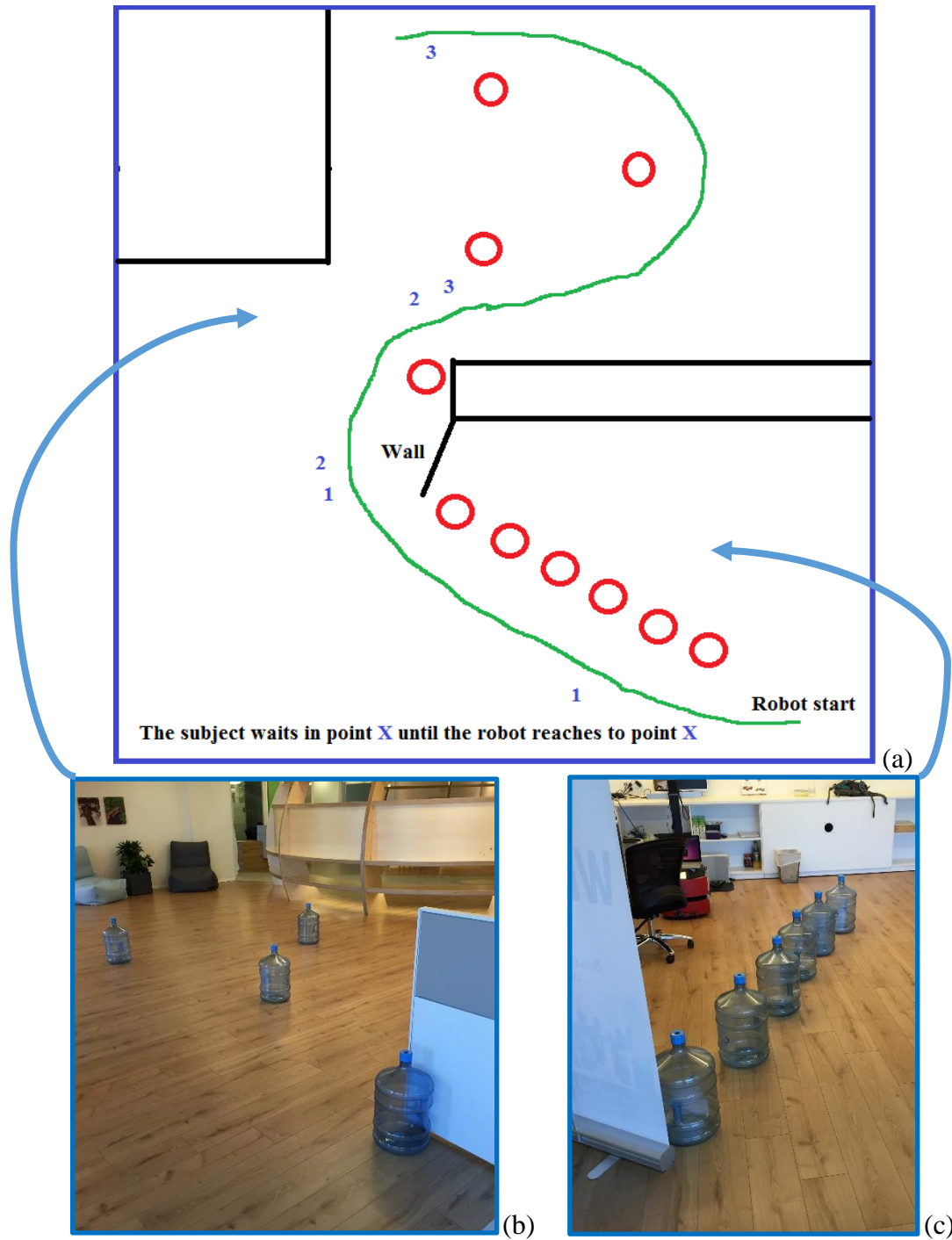


Figure 9 - Direct-Following and History-Following experiment path. (a)-schematic description. (b)-photograph of end of the path. (c)-photograph of start of the path

### 3.4.7 Adaptive Kinect-Laser vs. Non-Adaptive (for Direct Following and History Following) Experiment

The two human-following methods (*DF*, *HF*) with the Kinect V2 with the best combination of integrated algorithms (from the previous experiment, Section 3.4.6, denoted as Non-Adaptive methods) were compared to the same human-following methods with a laser legs detector, which was actuated when the Kinect lost the participant (denoted as Adaptive methods). This experiment also took place in the offices of the CDI in Beer-Sheva Israel and

included a 25-m length path with three obstacles, a wall, and a corner (**Figure 10**). At the beginning of the experiment, the subject was asked to stand in front of the robot so that the robot's Kinect and laser sensors could detect her/him. If the robot was not able to detect the legs of the subject because the width of her/his legs fell below the defined threshold for laser detection, the subject was asked to wear rain boots to 'widen' the legs. When the trial started, the subject walked slowly without stopping from the starting point through points 1 and 2 to point 3 (marked on the floor). Once subject arrived at point 3, s/he was asked to wait until the robot reached the first wall, and only then to complete the walking path. When the subject moved from point 3 to the end-point, s/he disappeared behind the wall to simulate the situation in which a person turns a corner in a hallway. Each subject completed this path four times with different combinations of algorithms (**Table 2**) and in a different order. Twenty-four participants participated in this experiment.

*Table 2- Combinations of methods and algorithms Adaptive vs. Non-Adaptive experiment*

Trial	Following method	Obstacle avoidance	Depth occlusion	Search after disappear	Laser legs' detector
<b>1</b>	History	√	√	√	√
<b>2</b>	History	√	√	√	X
<b>3</b>	Direct	√	√	√	√
<b>4</b>	Direct	√	√	√	X

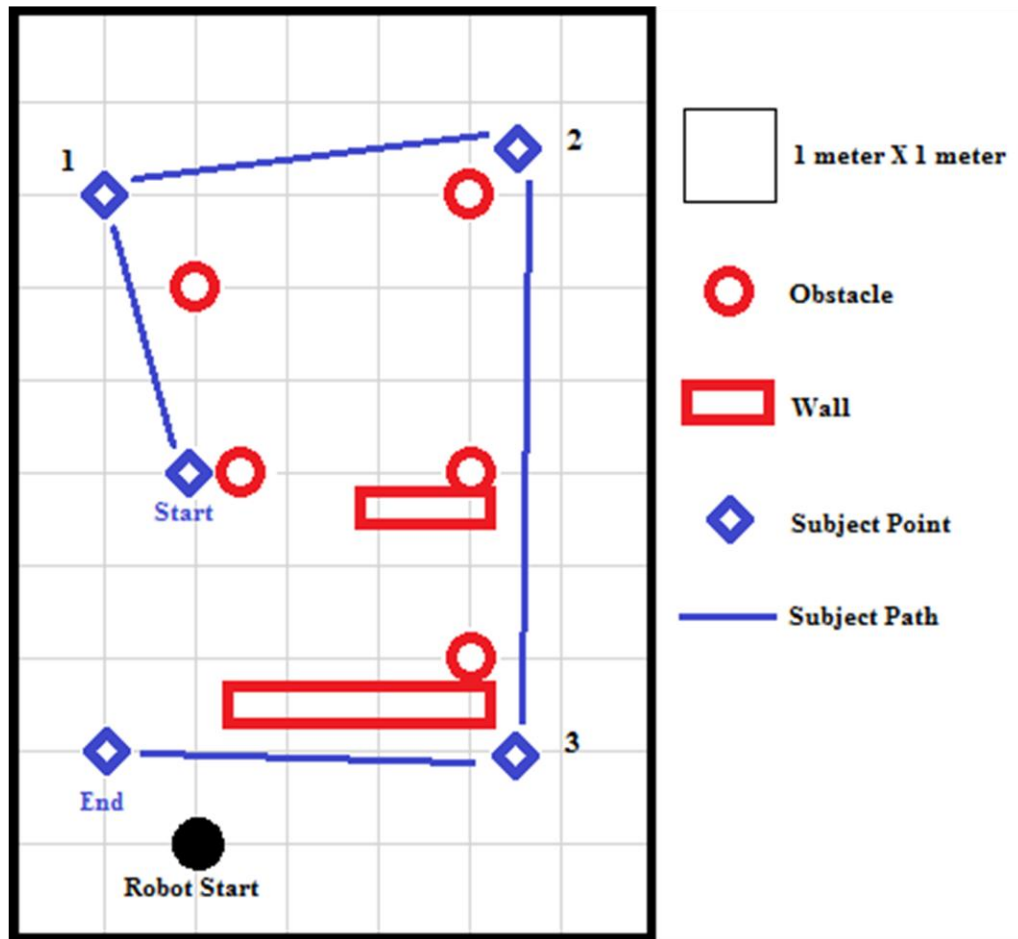


Figure 10- Adaptive vs. Non-Adaptive experimental paths. (a)-schematic description. (b)-photograph

### 3.5 ROS implementation

*DF* and *HF* methods and their integrated algorithms (*DO*, *OA*, and *SAD*) were implemented in ROS with the following nodes and topics, as shown in **Figure 11**.

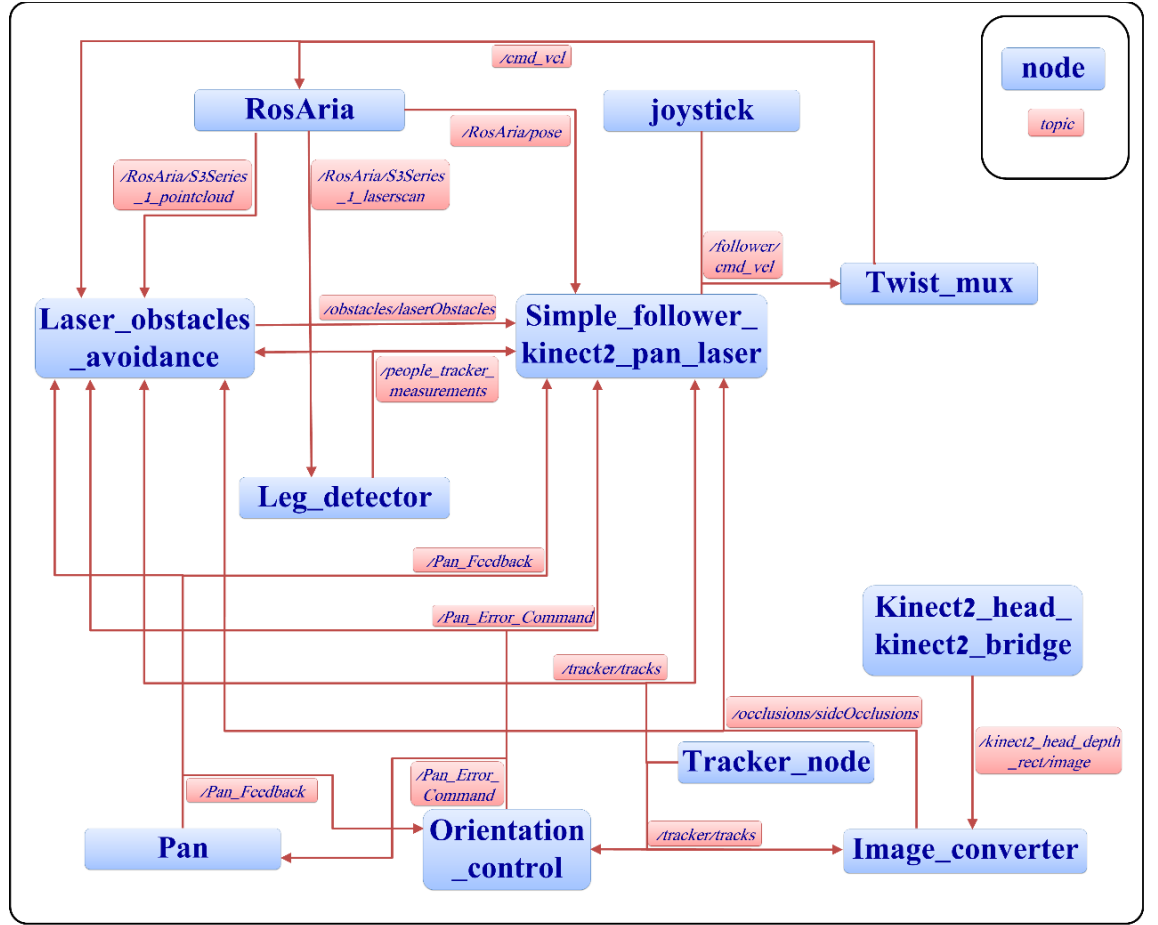


Figure 11- Nodes and topics flow-chart

- **RosAria** is the robot's main node. According to the movement of the robot's wheels, it calculates the robot's position and sends it to **Simple\_follower\_kinect2\_pan\_laser** by `/RosAria/pose`. It sends the laser scan in two ways: 1) to the **leg\_detector** by `/RosAria/S3Series_1_laserscan` and 2) to **laser\_obstacles\_avoidance** by `/RosAria/S3Series_1_pointcloud`. It is also responsible for the transformation of coordinated positions in the different sensors by `/tf`.
- **Twist\_mux** is the node that responsible for the subsequent commands to move the robot. It decides on the priority of the incoming nodes. It receives move commands from **Simple\_follower\_kinect2\_pan\_laser** and from **joystick** (via **turtlebot\_teleop\_joystick**). It can also receive commands from the **safety** node, which was not used in this case (**safety** is responsible for locking the robot's

wheels when the laser detects close obstacles). It publishes the move commands to **RosAria** by */cmd\_vel*.

- **Simple\_follower\_kinect2\_pan\_laser** is the main node that includes the method of following (*DF* or *HF*) and if necessary uses the *SAD* integrated algorithm. It publishes on */follower/cmd\_vel* the linear and angular velocity commands to **twist\_mux**. It receives the following information from seven different topics for analysis and synchronization:
  1. The position of the robot from **RosAria** by */RosAria/pose*.
  2. The position of an obstacle related to the robot from **laser\_obstacles\_avoidance** by */obstacles/laserObstacles*.
  3. The Boolean occlusion detection variables from **image\_converter** by */occlusions/sideOcclusions*.
  4. The position of the person by laser from **leg\_detector** by */people\_tracker\_measurements*.
  5. The position of the person by Kinect from **tracker\_node** by */tracker/tracks*.
  6. The position of the Pan related to the center of the robot from (**Pan**) **serial\_node** by */Pan\_Feedback*.
  7. The position of the person from the center of the Kinect from **orientation\_control** by */Pan\_Error\_Command*.
- **Leg\_detector** is the node that detects the position of a person's legs with the laser. It receives the robot's laser scan determined by the laser sensor from **RosAria** by */RosAria/S3Series\_1\_laserscan* and is responsible for the transformation between the laser measurements and the robot's position with */tf*. It publishes the coordinates of the person related to the robot to **Simple\_follower\_kinect2\_pan\_laser** by */people\_tracker\_measurements*.
- **Laser\_obstacles\_avoidance** is the node that searches with the laser for obstacles near the robot. It receives the robot's laser scan from **RosAria** by */RosAria/S3Series\_1\_pointcloud*, the linear and angular velocity of the robot from **Twist\_mux** by */cmd\_vel*, and all the parameters to calculate the position of the person with laser and Kinect (such as the information that **Simple\_follower\_kinect2\_pan\_laser** receives). It publishes the linear and angular velocity that is required to avoid collision if there is an obstacle to **Simple\_follower\_kinect2\_pan\_laser** by */obstacles/laserObstacles*.
- **Image\_converter** is the node that detects occlusions near the person with the Kinect Depth image. It receives the position of the person with the Kinect from **tracker\_node** by */tracker/tracks* and the Kinect depth image from



**kinect2\_head\_kinect2\_bridge** by */kinect2\_head\_depth\_rect/image*. It publishes the Booleans of the size and side of an occlusion detection to **Simple\_follower\_kinect2\_pan\_laser** by */occlusions/sideOcclusions*.

- **Orientation\_control** is the node that is responsible for moving the Pan mechanism and for sending angular positions. It receives the position of the person with the Kinect from **tracker\_node** by */tracker/track*s and the position of the Pan related to the center of the robot from **(Pan) serial\_node** by */Pan\_Feedback*. It publishes the angular velocity that the Pan needs in order to maintain the person in the center of the Kinect to **(Pan) serial\_node** by */Pan\_Error\_Command*.

## 3.6 Analysis

### 3.6.1 Performance Measures

The performance measures described below were used:

The following measures were counted manually during each trial:

- Number of losses of the person
- Number of self-recoveries and percent of self-recoveries out of total losses
- Number of interventions due to losses and percent of these interventions out of total losses
- Number of safety interventions
- Number of Kinect collapses
- Number of collisions with obstacles

The following measures were calculated directly from the ROS procedures:

- Number of laser detections of obstacles
- Distance between the robot and the participant (average and standard deviation)
- Length of the robot's path
- Number of matches between the Kinect person-detection position and laser legs detector position that were <20 cm
- Ratio between tracking and no tracking from the Kinect and from the laser separately.

Each trial was recorded by Rviz and rqt\_console to calculate:

- The participant's walking velocity
- The percent of false alarms for the position of the legs
- The *DO* algorithm's true positive and false alarm (false positive) results

The velocity of the participant was calculated by dividing the total time of subject walking by the total path length s/he walked. The percent of legs false alarm detection was also calculated from the recording of the trial by counting the time of true legs detection and the time of false alarms. The true positives and false alarms of the **DO** algorithm were calculated by using the ROS information and the recording. The relative time according to the position of the robot related to the position of the participant was derived manually from the recording and compared to the actual ROS information.

### 3.6.2 Statistical Analyses

Statistical analyses included SPSS ANOVA with 0.05 confidence level. To test for a significant difference between more than two trials, a post-hoc pairwise comparison (Tukey test) was conducted (**Table 3**). All raw data and the statistical analyses are detailed in Appendix C.

*Table 3- Statistical analyses*

Experiment	Section	Hypothesis	Statistical tests
<b>Comparison of occlusion algorithms</b>	<b>3.4.5</b>	Depth – preferable close distance; Vision – preferable far distance	ANOVA and Pearson - confidence level of 0.05
<b>Direct-Following and History-Following</b>	<b>3.4.6</b>	Combination of all the integrated algorithms in each method is the best	ANOVA and Tukey - confidence level of 0.05
<b>Adaptive Kinect-Laser Direct Following and History Following</b>	<b>3.4.7</b>	Adaptive methods are better than Direct methods;	ANOVA and Tukey - confidence level of 0.05

## 4. Chapter Four: Algorithms

### 4.1 Overview

To summarize the previous chapters, two main methods for human-following were improved and implemented on the human-following robot:

- *Direct-Following (DF)*
- *History-Following (HF)*

Three integrated occlusion detection algorithms were developed for these methods:

- *Depth-Occlusions-Detection (DO)*
- *Vision-Occlusions-Detection (VO)*
- *Combined-Occlusions-Detection (CO)*

Two integrated algorithms were developed for obstacle avoidance and search after disappearance:

- *Obstacle-Avoidance (OA)*
- *Search-after-Disappear (SAD)*

All C++ codes are shown in Appendix D.

Algorithm	Innovations
<b>General</b>	<ul style="list-style-type: none"><li>• All the algorithms and following methods work without a-priori information about the environment or any kind of pre-built map of the environment</li></ul>
<b>Depth-Occlusions (DO)</b>	<ul style="list-style-type: none"><li>• Real-time occlusion detection using depth information of the pixels</li><li>• Compares the depth value of the pixels (distance value of the pixels) inside the BBC of a detected person to the distance of the whole person from the robot and searches for small values that indicate closer pixels (indicating an occlusion)</li><li>• Reduces false alarms and can detect both small and large occlusions and even a vertical occlusion like a wall</li><li>• Change the following angle to increase the line of sight</li></ul>
<b>Obstacles-Avoidance (OA)</b>	<ul style="list-style-type: none"><li>• Real-time obstacle detection and avoidance</li><li>• Declares an adaptive corridor in front and on the sides of the robot to narrow the scan area, which depends on the prevailing linear and angular velocities of the robot</li><li>• Search for obstacles inside the turning radius</li></ul>

<b>Search- After- Disappear (SAD)</b>	<ul style="list-style-type: none"> <li>• Increase of the ability to refollowing</li> <li>• Moves the robot to the person's last known position and turns the robot in the direction that is calculated according to the last few frames obtained before the person had disappeared</li> </ul>
<b>Direct- Following (DF)</b>	<ul style="list-style-type: none"> <li>• Moves the robot directly to the position of the detected person</li> <li>• Gives priority to sending the robot the linear and angular velocity to avoid any obstacles change</li> </ul>
<b>History- Following (HF)</b>	<ul style="list-style-type: none"> <li>• Moves the robot directly to the historical position of the person being tracked</li> <li>• avoids big changes of the position of the person caused by the movements of the robot and the Kinect</li> <li>• avoids quick turns that cause the Kinect to lose the person</li> </ul>

## 4.2 Depth Occlusions Detection

The *DO* detection algorithm *compares the depth value of pixels inside the BBC of a detected person to the distance of the whole person from the robot and searches for small values that indicate closer pixels (indicating an occlusion)*. The number of pixels that are closer than the distance of the person from the robot is counted by adding a threshold to reduce small distance measurement errors and to avoid other body parts detected as closer pixels.

In order to reduce false detections of a person, a valid tracking of a person is indicated only if there is the tracking passes three thresholds of confidence within minimum and maximum heights of the person (ConfidenceTheshold, HeightTheshold, HeightMaxTheshold).

The OpenPTrack (Munaro et al. 2014) provides four parameters of the BBC around the detected person. The *DO* refers to them as:  $x_{min}$  the X value of the top-left corner of the BBC,  $y_{min}$  the Y value of the top-left corner of the BBC,  $x_{max}$  the X value of the top-right corner of the BBC and  $y_{max}$  the Y value of the bottom-right corner of the BBC. In addition, the center X value of the BBC is also calculated ( $x_c = (x_{max} + x_{min})/2$ ).

Since the detection parameters are related to the size of the BBC, we incorporated two changes, as follows (**Figure 12**). First, to *avoid the ground depth value and to reduce false alarms*, the 1/8 lower part of the BBC was cut ( $down_{cut} = round((y_{max} - y_{min})/8)$ ). Second, to add *dependency on the person's distance from the robot from the width of the BBC*, a margin was added to the BBC ( $margin_{add} = round(10/distance)$ ). The margin

was added also to predict an occlusion process before it happens. After these changes, the new BBC was ready for the occlusions detection process.

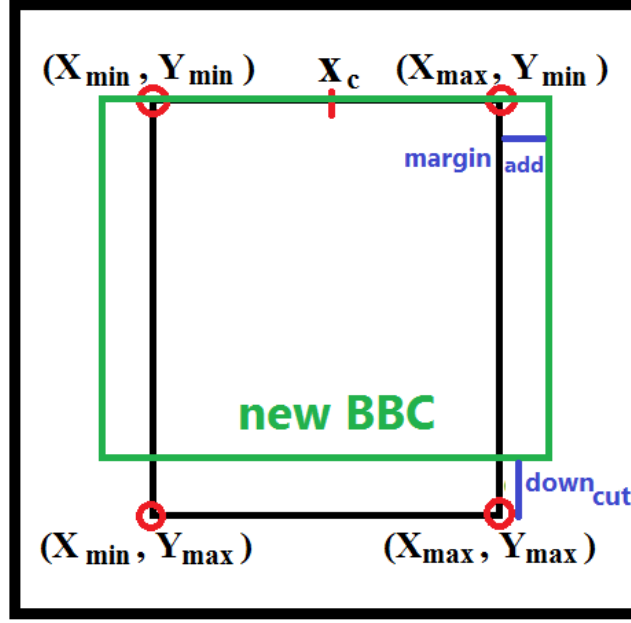


Figure 12- New BBC parameters (DO)

Each depth pixel value was normalized to 0-255 and compared to the normalized distance of the person from the robot. To **prevent small measurement errors**, a depth constant threshold was added to detect depth values of closer pixels ( $DEPTH_{THRESHOLD} = 3.0$ ). The values of the counters were increased for closer distances by adding a threshold of 3 (equal to a distance of 0.5 m). For the left side, the pixels lie between  $[x_{min} - margin_{add}, x_c - 5]$  (from the left side of the BBC adding a small margin to the center of the BBC without the last 5 columns). For the right side, the pixels lie between  $[x_c + 5, x_{max} + margin_{add}]$  (from the center of the BBC without 5 first columns to right side of the BBC, adding a small margin). For **wall detection**, all the values of the same column must contain a smaller depth value (indicating an occlusion from the top to the bottom of the BBC).

To declare a small or large occlusion from the left or the right, the BBC must be covered by  $\frac{1}{3}$  to a  $\frac{1}{2}$  of closer pixel values for small occlusions and  $>\frac{1}{2}$  for large occlusions.

$$smallOcclusion_{left/right} = round\left(\left(\frac{x_c - x_{min}}{3}\right) * (y_{max} - y_{min}) * \frac{7}{8}\right)$$

$$bigOcclusion_{left/right} = round\left(\left(\frac{x_c - x_{min}}{2}\right) * (y_{max} - y_{min}) * \frac{7}{8}\right)$$

In addition, for wall detection, the algorithm finds continuous occlusions from the left or right by finding entire columns in the BBC that have depth values that are closer than the person's distance to the robot. There are four counters, one for each side for small and large occlusions ( $left_{count}, right_{count}$ ), and one for each side for wall detection

( $leftWall_{count}, rightWall_{count}$ ). Occlusion detection is published using six Boolean variables (three for the left side and three for the right side) for small, large and wall detections: ( $smallLeft_{true}, bigLeft_{true}, wallLeft_{true}$ ), ( $smallRight_{true}, bigRight_{true}, wallRight_{true}$ ).

A pseudo code is shown in **Figure 13** and an example of the *DO* algorithm in **Figure 14**.

**DO Pseudo code:**

- A. Input: the parameters and BBC of a detect person (with thresholds)
- B. Add margin to the BBC from left and right depend on the distance
- C. Ignore the 1/8 lower part of the BBC
- D. Left side (the same idea with right side): Initialize countLeft=0;
- E. For each left side pixel of the new BBC:
  - i. Initialize each column countLeftWall=0;
  - ii. Compare the distance to the depth value
  - iii. If the value is smaller (threshold) than countLeftt++ and countLeftWall++
- F. Conditions:
  - i. If countLeftWall= number of rows in the new BBC, than LeftWall=true
  - ii. If countLeft between smallLeftOcclusions and bigLeftOcclusions than smallLeftOcclusions=true
  - iii. If countLeft bigger than bigLeftOcclusions than bigLeftOcclusions=true
- G. Output: 6 occlusions Booleans (smallRightOcclusions, smallLeftOcclusions, bigRightOcclusions, bigLeftOcclusions, RightWall, LefttWall)

*Figure 13 - DO pseudo code*

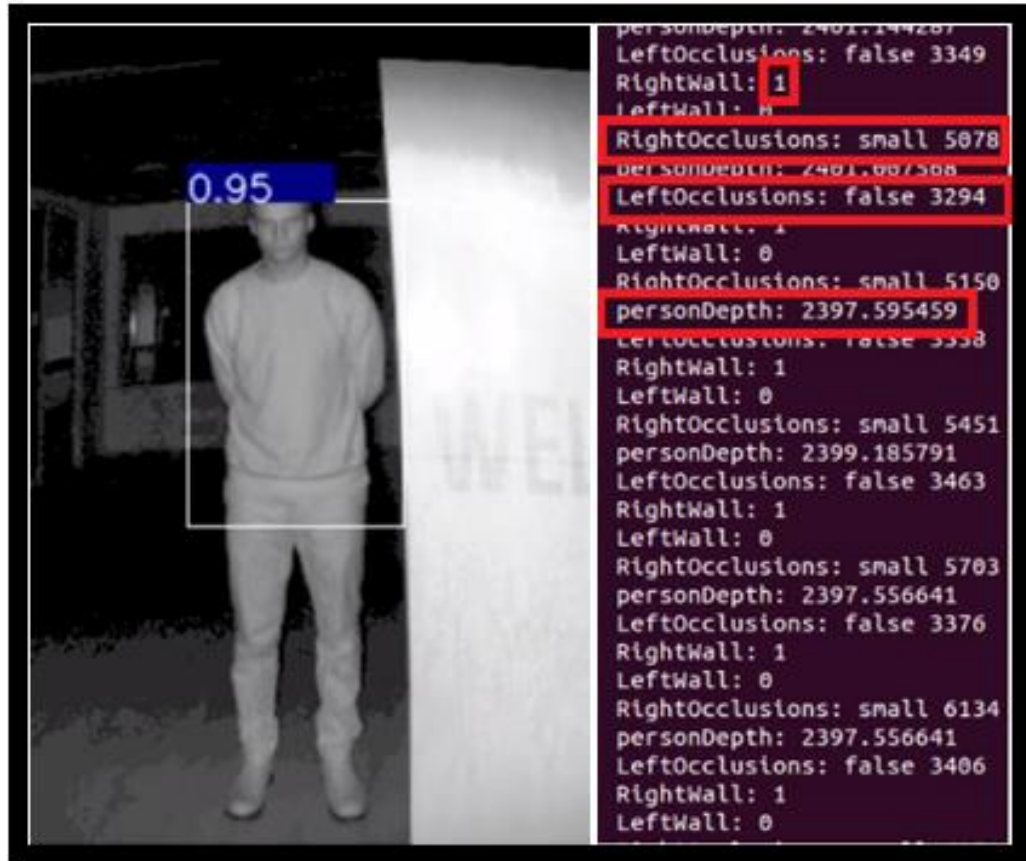


Figure 14 - DO person detection with a half-BBC (from OpenPTrack) near a right wall (left). Person's Distance of 2.4 m with a small right occlusion and right wall detection without a left occlusion (right)

### 4.3 Vision Occlusions Detection

The **VO** algorithm uses the ROI of the detected person from a depth image and fuses it with a MONO image from the Kinect to *detect occlusions in a 2D image*. In the MONO image, after basic image processing routines, the algorithm searches for straight vertical lines to detect a wall occlusion.

To reduce false detections from OpenPTrack, a tracking is indicated only if the tracking passes the same three thresholds as for the **DO**. The BBC parameters from OpenPTrack depend on the resolution of the Kinect depth image. The MONO image has a higher resolution; therefore, extra parameters were added to the original BBC parameters. Another problem with which the algorithm deals is the different horizontal fields of view between the MONO image and the depth image of the Kinect. Several steps were therefore undertaken to *match the pixel coordinates of the depth image to the MONO image*. These steps were needed because it is not simple to convert, resize and extract ROIs in the MONO image from the BBC parameters of the depth image due to the different resolutions [the resolution of the MONO image is twice that of the depth image (1920×1080)], and the two

techniques have different horizontal fields of view—MONO 84.1° and depth 70.6°]. The first step was to subtract 105 pixels from each side of the MONO image to eliminate different right and left image borders. Then, the image was resized to the same resolution as that of the depth image (960×540). Another problem caused by the differences in the horizontal field of view was an unequal X value between the  $x_c$  value that was calculated from the BBC of the depth image and the real position of the person in the MONO image. This error increases as the distance increases due to the different convexities of the depth and MONO images. The best solution was to add pixels to the BBC from the left side of the ROI, depending on the distance from the center of the image of the X-axis (the center of the image is X=270) divided by 3:

$$monoX_{min} = x_{min} * 2 + round\left(\left(\frac{270 - x_c}{3}\right) - distance * 2\right)$$

For the right side, depending on the width of the original depth BBC and the new  $monoX_{min}$  that was calculated, a new right side was declared:

$$monoX_{max} = monoX_{min} + (x_{max} - x_{min}) * 1.4$$

The right side of the ROI in the MONO image was expanded to cover the whole person by multiplying the width from the depth image by 1.4.

For the Y-axis new coordinates, the same minimum was taken, namely,  $monoY_{min} = y_{min}$  and for the maximum value, the height of the ROI was multiplied only by 1.3 to include the legs but not the ground:

$$monoY_{max} = monoY_{min} + (y_{max} - y_{min}) * 1.3$$

OpenCV was used for the image processing on the ROI, with the following steps and parameters, which were empirically derived: Gaussian blur of size 3×3, Canny edge detector with a low threshold of 50 and high threshold of 300 with a Sobel 2 sized and an L1 norm, opening and then closing the pixels by 5×5.

To **find the contours of the whole person**, the OpenCV `findContours` function was used. The function includes the `contours` vector of vector of points for saving the contours, with a retrieval mode that organizes the contours into a two-level hierarchy and compresses horizontal, vertical, and diagonal segments and leaves only their end points. To avoid small contours, the size of the contour was compared to the height of the ROI. Only if the contour was bigger than the whole height of the ROI multiplied by 1.5, was it noted as belonging to the ROI; otherwise, it was deleted from the vector `contours`.



*Straight lines* were derived using the OpenCV HoughLinesP function. The function includes vector **lines** that contain 4 arguments ( $x_{start}, x_{end}, y_{start}, y_{end}$ ) for each line, with a resolution of 1 pixel and  $1^\circ$ , with a minimum threshold of 50 and with a minimum line length of half of the ROI height. To prevent too many straight lines in the ROI, only vertical lines that were less than 1/10 size of the person's box width were derived. If the edges of the straight line were inside the ROI from the left to the center minus 5, the line was defined as a left wall and if the edges of the straight line were inside the ROI from the center plus 5 to the right, it is defined as a right wall.

The pseudo code of the algorithm is shown in **Figure 15** and an example of the **VO** algorithm in **Figure 16**.

VO Pseudo code:

- A. Input: the parameters and BBC of a detect person (with thresholds)
- B. Convert, resize and extract ROI coordinates in MONO from the original BBC
- C. Gaussian Blur (3\*3), Canny edge detector, dilate (5\*5), erode (5\*5)
- D. Find contours that are bigger than 1.5 the height of the ROI
- E. Find near vertical straight lines (0.1 width of the ROI) that are bigger than half of the ROI
- F. Wall detection:
  - i. If the xStart and xEnd values of the line are inside the ROI between the left to the center than LeftWall=true
  - ii. If the xStart and xEnd values of the line are inside the ROI between the center to the right than RightWall=true
- G. Output: 2 occlusions Booleans (LeftWall, RightWall)

*Figure 15 - VO pseudo code*



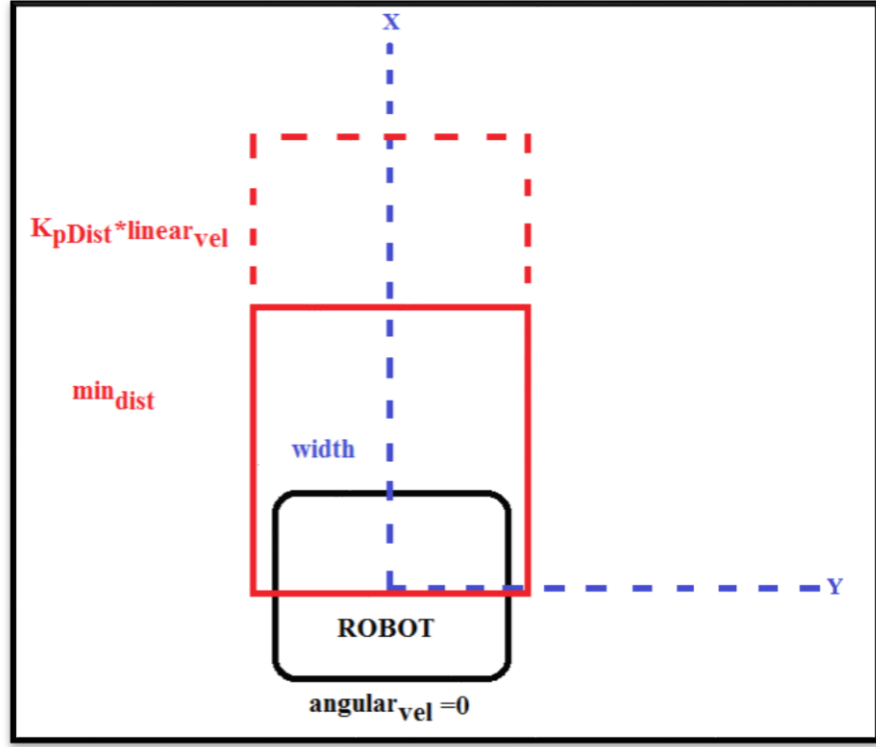
and hence the obstacles detection algorithm is ignored to prevent confusion between the person and the obstacle.

The corridor that may contain obstacles depends on the linear and angular velocity of the robot (**Figure 17**). For the X-axis (in front of the robot), the minimum distance to search for obstacles is  $min_{dist} = 0.8$ , irrespective of the robot's linear velocity. In other words, if there is a point that the laser detects that is  $<0.8$  m in front of the robot, the robot declares it as an obstacle, independent of robot's velocity. At greater distances ( $>0.8$  m), the declaration of obstacles depends on the linear velocity of the robot multiplied by  $K_{pDist} = 3$ . For example, if the linear velocity of the robot is 0.4 m/s, then an obstacle can be found up to  $0.4 \times 3 = 1.2$  m in front of the robot. In addition, all the obstacles detected by the laser or Kinect sensors that are near the detected person, namely, in a 1 m radius of the person ( $RADIUS_{PERSON} = 1.0$ ), are ignored. To search in the negative X-axis if the robot is turning, the absolute value of the angular velocity is used. If the angular velocity is large, the algorithm searches for larger negative values of X.

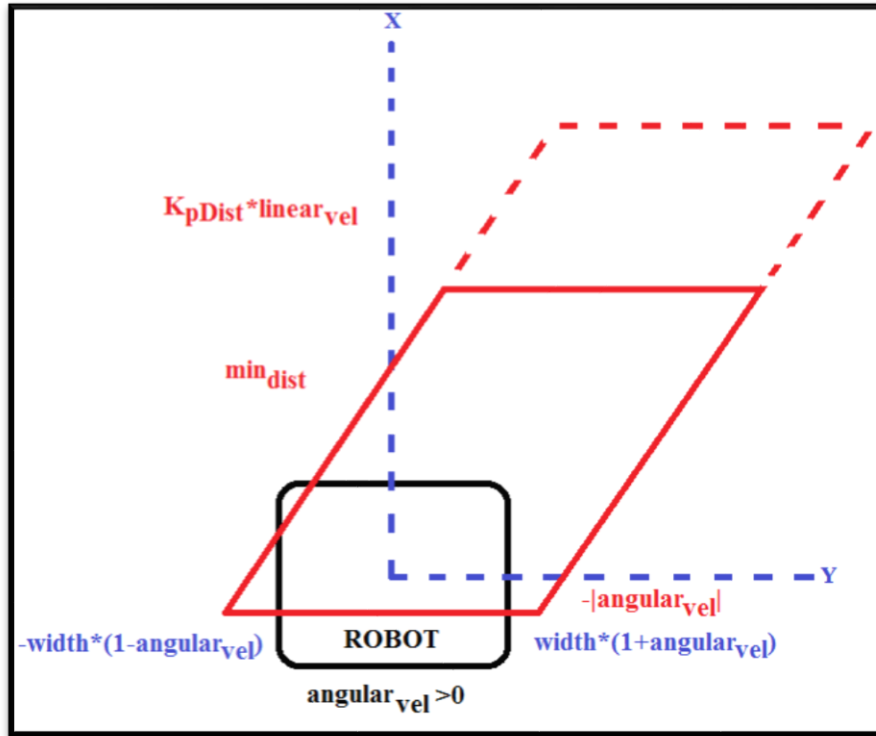
If  $((x_{obstacle}[i] < K_{pDist} * linear_{vel}) \cup (x_{obstacle}[i] < min_{dist})) \cap (x_{obstacle}[i] > -|angular_{vel}|) \cap ((\sqrt{(x_{obstacle}[i] - x_{person})^2 + (y_{obstacle}[i] - y_{person})^2}) > RADIUS_{PERSON})$

For the Y-axis (the sides of the robot), the obstacle must lie between the  $width = 0.5$  multiplied by  $1 + angular_{vel}$  and  $-width$  multiplied by  $1 - angular_{vel}$ . If the angular velocity is zero, then the corridor lies between 0.5 and -0.5, but if the robot is turning and the angular velocity is not zero, then the corridor moves to the side of the turn in order **to search for obstacles inside the turning radius**. As for the X-axis, the technique to ignore obstacles near the person is used.

If  $((y_{obstacle}[i] < width * (1 + angular_{vel})) \cap (y_{obstacle}[i] > -width * (1 - angular_{vel})))$



(a)



(b)

Figure 17 - Obstacles corridor examples. (a)-robot moves only forward.  
(b)-robot turns right.

The next step is to choose the closest obstacle to the robot inside the corridor. For each point that is declared an obstacle inside the corridor, the distance of the point to the robot is calculated as:  $dist_{obstacle}[i] = \sqrt{x_{obstacle}[i]^2 + y_{obstacle}[i]^2}$ . The closet point coordinates are passed to the next step and declared as an obstacle.

The final step is to calculate and publish the robot's linear and angular velocity to the robot so that it can avoid the closet obstacle. For the linear velocity, after many trials with different linear velocity values, a 0.2 m/s was chosen. For the angular velocity, the decision depends on the position of the obstacles in the Y-axis related to the robot (from the left side of the robot or from the right side). If the obstacle is to the robot's left, a command to turn right is issued (negative angular velocity). *The size of the angular velocity depends on the distance from the width value* (0.5) divided by 2. If the obstacle's Y coordinate is at the center of the robot (i.e., equal to zero), then the angular velocity is large. If the obstacle is at the robot's left, then:

$$angular_{command} = -(width - y_{closest})/2.$$

If the obstacle is at the robot's right, a command to turn left is issued (positive angular velocity). The size of the angular velocity depends on the distance from the  $-width$  value (-0.5) divided by 2. If the obstacle's Y coordinate is at the center of the robot (i.e., equal to zero), then the angular velocity is large. If the obstacle is to the robot's right, then:

$$angular_{command} = (-width + y_{closest})/2.$$

#### 4.6 Search-after-Disappear

In the **SAD** algorithm, the robot remembers the last position of the person and subtracts between two values of  $y_{kinect}$ , namely, the last one and the value from four frames previously. A positive  $y_{kinect}$  implies that the person is on the right side of the robot, and a negative  $y_{kinect}$ , that the person is on the left side. The value of the subtraction is defined as *the direction of turning of the robot* in order to search for the person. After performing these calculations, the robot moves to the last position of the person with constant linear velocity of 0.3 m/s for a duration that depends on the distance of the last detection. When it reaches this last position, it turns in the direction that it had calculated from the last four frames of detection.

#### 4.7 Kinect Orientation Control – Pan Mechanism

The Pan mechanism moves to maintain the person being detected in the center of the Kinect. Two levels of code were employed. The high-level code is an ordinary ROS implementation with topics, publishers and subscribers. The low-level code, developed by Doisy and co-workers, controls the movement commands of the Pan (Doisy et al. 2012).

The high-level code "Kinect\_orientation\_control" uses OpenPTrack to detect people with three thresholds (confidence, height and max height). The code subscribes from two topics: OpenPTrack parameters using `/tracker/tracks` and the angle of the Pan related to the center of the robot using `/Pan_Feedback` that is published from the low-level code. The code publishes the angle error of the Pan to the low-level code to angle the Pan to maintain the person in the center of the Kinect using `/Pan_Error_Command`. After receiving the parameters of the person from OpenPTrack and passing the thresholds, the code calculates the angle of the person from the center of the Kinect as  $\tan^{-1}\left(\frac{y_{kinect}}{x_{kinect}}\right)$  of the person. The Y-axis is from left to right of the Kinect (zero means the center of the Kinect) and the X-axis is the depth (distance from the Kinect). The angle of the person from the center of the Kinect is passed to the low-level code to move the Pan until the angle is zero (y equal to zero). If no person is detected for more than three seconds, the command that passes to the low-level code is half of the angle of the Pan related to the center of the robot in the opposite direction (`error_command.data=-0.5*AngleErrorPan;`); this command causes the *Pan to return to the center of the robot*.

The low-level code subscribes from two topics: 1) the move command from the high-level code using `/Pan_Error_Command` and 2) a topic that sends true or false to the Pan for the actual move command, using `/Start_Stop_Pan`. It publishes the angle of the Pan related to the center of the robot using `/Pan_Feedback` to the high-level code.

The parameters for the movement of the Pan were empirically selected in a series of trials. On the one hand, the Pan must move fast enough not to lose the person. On the other hand, it must not move too fast to overshoot the person and move back and forth all the time because of fast movements (like a harmonic motion). The best parameters were derived to 0.5 max speed of movement to avoid overshooting. Additionally, a small threshold of 0.01 rad was added to prevent small movements of the robot when the Pan is near the center of the robot.

## 4.8 Direct-Following Method

This method of human-following by a robot causes the robot to *move directly to the position of the detected person*. This method transforms and calculates the position of the person obtained by the Kinect and by the laser, sends commands to the robot according to the *OA*, changes the following angle according to the occlusions detection algorithms (*DO*, *VO*, *CO*), and compares the results with and without *SAD*.

The integrated **OA** algorithm receives the Boolean values for slowing down and obstacle detection and it also receives the values for the linear and angular velocities of the robot directly from the **OA** algorithm. The first *priority of sending the robot it's linear and angular velocities to enable it to avoid any obstacles* is implemented in all sections of this main method. If the **OA** detects an obstacle, then the robot moves according to the position of the obstacle in order to avoid it. Only when the **OA** detects that the path is clear, does the method continue to send the commands to the robot to follow the person. During the movements of the robot relative to the obstacle, the Kinect with the Pan mechanism and the laser leg detector continue to follow the person without sending commands to the robot to move. If **OA** is not running, then the method notes that there are no obstacles.

For the integrated occlusions detection algorithms (**DO**, **VO**, **CO**), The **DF** method receives all the Boolean values for large and small occlusions from the right and the left. It *changes the following angle* to 15° for small occlusions, and to 30° for large occlusions. When the occlusions detection algorithms are not running, the following angle equals zero (the robot moves directly to the person being detected).

The integrated **SAD** algorithm compares the last position of the person before losing her/him with several frames before to realize the person's drift. To achieve this, the X- and Y-axes of the person detected by the Kinect must be transformed to the robot's position. From the Kinect detection, the transformation uses three variables that are related to the position of the person vis-à-vis the robot, namely,  $a_k$  = the angle in radians of the person related to the center of the Kinect,  $a_p$  = the position of the Pan related to the center of the robot in radians,  $D_k$  = the distance of the person from the Kinect. If there is a person detected by the Kinect (OpenPTrack), then her/his position in relation to that of the robot is defined as:

$$x_{kinect} = D_k * \cos(a_k + a_p)$$

$$y_{kinect} = D_k * \sin(a_k + a_p)$$

For **DF**, the method uses many parameters, variables and constants. To calculate the distance ( $D$ ) and angle ( $a$ ) of the person in relation to the robot, the **DF** method uses the position of the person as obtained from the Kinect and from the laser detector. It sends velocity commands with upper bounds of  $max_{speed} = 0.3$  and  $max_{turn} = 0.2$ , and with linear and angular speed controller  $K_{pDist} = 0.2$ ,  $K_{pAngle} = 0.5$ , respectively. A constant of the distance from the person  $D_{TARGET} = 1.2$  is also included. The last parameter that depends on the integrated occlusions detection algorithm (**DO**, **VO**, **CO**) is the value of the  $a_{following}$  that by default equals zero if no occlusion has been detected.

$$D_{kinect/laser} = \sqrt{x_{kinect/laser}^2 + y_{kinect/laser}^2}$$

$$a_{kinect/laser} = \tan^{-1} \frac{y_{kinect/laser}}{x_{kinect/laser}}$$

$$angular_{command} = \max((a_{kinect/laser} + a_{following}) * K_{pAngle}, max_{turn})$$

$$linear_{command} = \max((D_{kinect/laser} - D_{TARGET}) * K_{pDistance}, max_{speed})$$

The method first uses the Kinect detection, but if it loses tracking, it then changes to the laser detection algorithm. If there is no detection by both sensors, it uses the **SAD** integrated algorithm. This method can even work with only one source of person detection (Kinect or laser). A method that works with both of the sources together will be described in Section 4.10 'Adaptive Following Methods'.

## 4.9 History-Following Method

The robot is able to avoid obstacles without detecting them by moving in the same path that the person walks; of course, as long as the person does not jump over an obstacle. A semi path follower that uses the history positions of the person and ***moves the robot directly to these historical points*** was developed. The X- and Y-axes of the detected person obtained from the Kinect or the from the laser detector were transformed to ***world coordinates*** and related to the position of the robot in the world. This transformation was done without a map of the environment and it can lose stability after a while due to robot slips. To calculate the position of the person in relation to the world, the position and orientation of the robot in relation to the world was obtained from RosAria ( $x_{robot}, y_{robot}, o_{robot}$ ).

For the laser calculation (**Figure 18**), the method first calculates the distance  $D_{laser}$  and angle  $a_{laser}$  of the person in relation to the robot according to the laser:

$$D_{laser} = \sqrt{x_{laser}^2 + y_{laser}^2}$$

$$a_{laser} = \tan \frac{y_{laser}}{x_{laser}}$$

Then, it transforms the position of the person to the world coordinates:

$$x_{laserPath} = x_{robot} + \cos(o_{robot} + a_{laser}) * D_{laser}$$

$$y_{laserPath} = y_{robot} + \sin(o_{robot} + a_{laser}) * D_{laser}$$



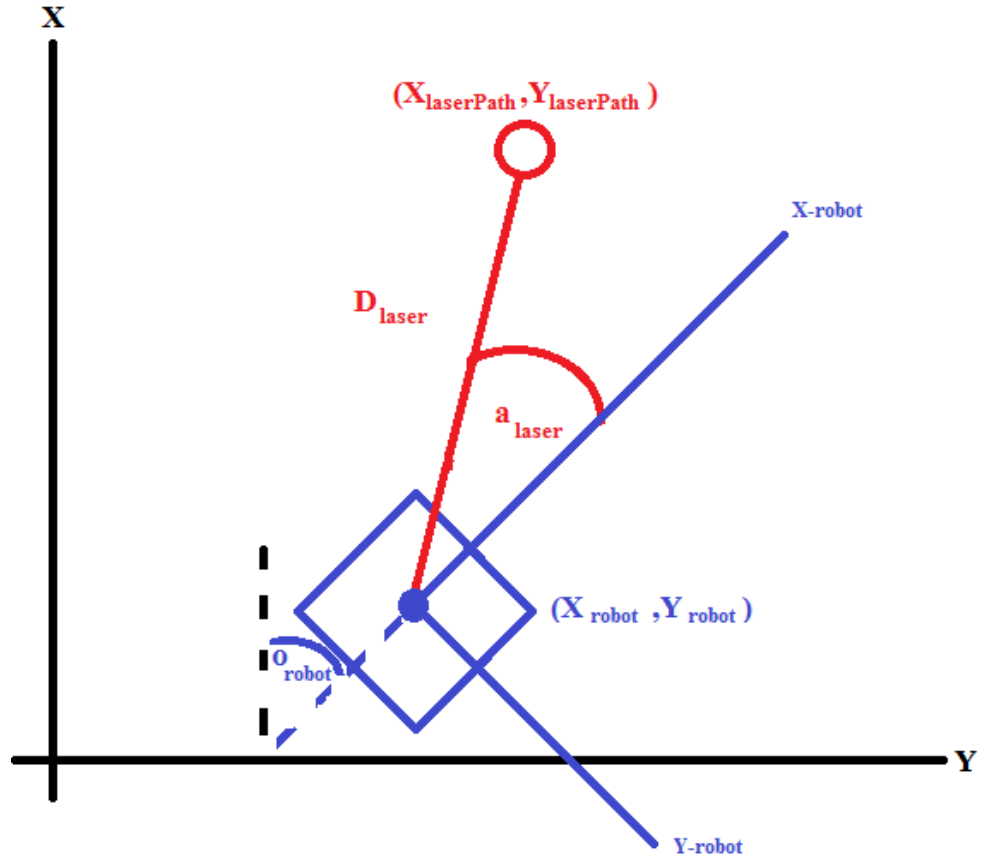


Figure 18 - Laser to world coordinates

For the Kinect calculation (**Figure 19**), the method uses three parameters:  $a_k = \text{AngleSmallError}$ , the person related to the center of the Kinect (from `/Pan_Error_Command`);  $a_p = \text{AngleErrorPan}$ , the position of the Pan related to the center of the robot (from `/Pan_Feedback`); and  $D_k = \text{msg} \rightarrow \text{tracks}[i].\text{distance}$ , the distance of the person from the Kinect (from `/tracker/tracks`).

The transformation of the person's position to the world coordinates is given by:

$$x_{kinectPath} = x_{robot} + \cos(\theta_{robot} + a_k + a_p) * D_k$$

$$y_{kinectPath} = y_{robot} + \sin(\theta_{robot} + a_k + a_p) * D_k$$

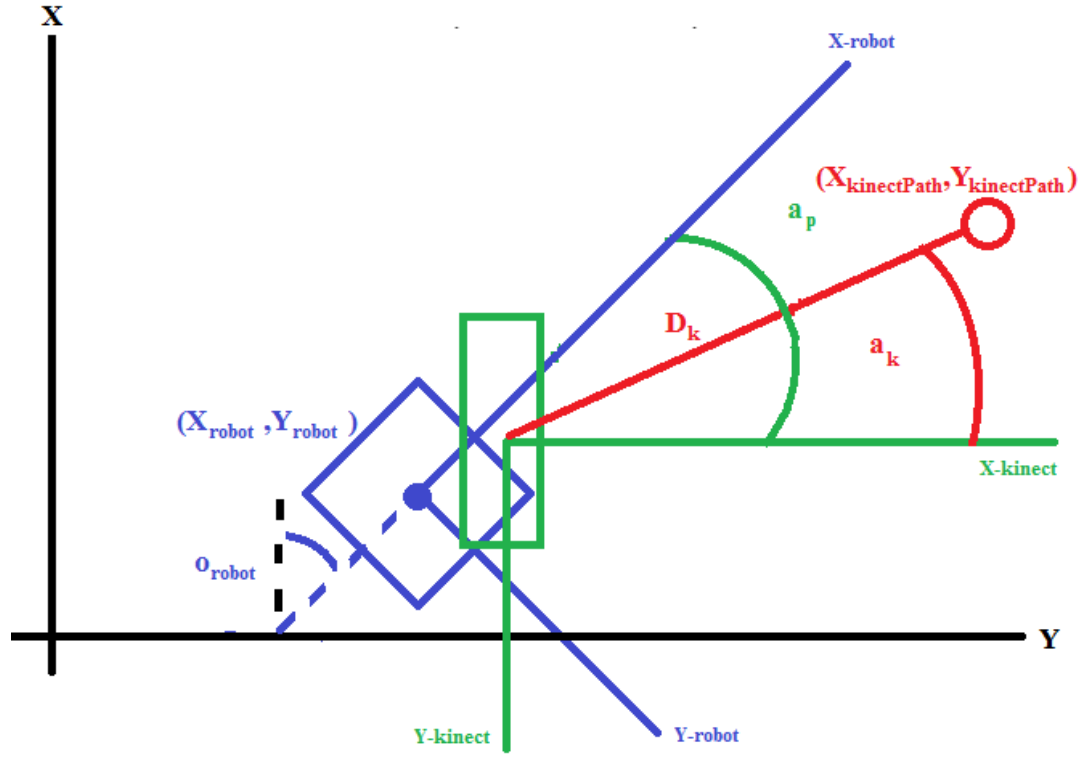


Figure 19 - Kinect to world coordinates

For using the history position, a vector of X values and Y values was inserted into the *HF* method to calculate the position of the person  $(x_{path}, y_{path})$ . The laser has an average frequency of 8 Hz, and the Kinect has an average frequency of 15 Hz. In order for the robot to use the same history points of the detected person, the 32<sup>nd</sup> point from the laser and the 60<sup>th</sup> point from the Kinect were taken each time to calculate the history position of the person (4 seconds of history). To calculate the angle ( $a_{follow}$ ) and the distance of following ( $D_{follow}$ ), these history points are used with the position and orientation of the robot, as follows:

$$a_{follow} = o_{robot} + \tan^{-1} \left( \frac{y_{follow} - y_{robot}}{x_{follow} - x_{robot}} \right)$$

$$D_{follow} = \sqrt{(x_{follow} - x_{robot})^2 + (y_{follow} - y_{robot})^2}$$

To **avoid problems near the forward and backward of the robot** caused by large differences when switching between positive and negative angle values, a positive transformation was added to the value of the angles.

If the absolute value between the orientation of the robot and the angle of the history following person to the robot is  $>\pi$  (3.14, half of a circle), then depending on whether the angle is positive or negative, the angle changes with  $2\times\pi$  (a circle) to avoid a large change near those values. In other words, the values for  $a_{follow}$  lie between  $[-\pi, +\pi]$ .

The velocity commands are executed in a manner similar to the *DF* method, but are related to the history point and the real time position of the person to prevent the robot from getting too close to the actual position of the person.

The angular velocity of the robot depends on  $K_{pAngle}=0.5$  (the twist speed controller) and on the angle of following:

$$velocity_{angular} = -a_{follow} * K_{pAngle}$$

If  $D_{kinect/laser} > D_{TARGET}$ , then the actual measurement of the distance from the Kinect or from the laser to the person exceeds the distance that the robot needs to maintain following. In such a case, the robot must move in such a way as to reduce the distance from the person, as follows:

$$velocity_{linear} = (D_{follow} - D_{TARGET}) * K_{pDistance}$$

Otherwise,  $velocity_{linear} = 0$

This equation implies that if the distance of the robot from the target (constant 1.2 m) exceeds the distance of the robot to the history point, then the robot's linear speed will be zero, indicating that the person is too close.

To ***prevent large changes in the position of the person caused by the movements of the robot and the Kinect***, a threshold of comparing the following samples was added and if the distance between two followers samples of person's position is  $>1$  m, the last sample will be ignored.

To ***avoid quick turns that cause the Pan to lose the person***, the angular velocity of the robot was limited to 0.5 rad/s (like the maximum speed of the Pan).

To ***avoid problems related to two sources of person detection***, the method uses first the Kinect detection, and if it loses tracking, then it changes to laser detection. If there is no detection by both sensors, the method can use the *SAD* integrated algorithm. This method can work even with only one source of person detection (Kinect or laser). A method that works with both of the sources together is described In Section 4.10 'Adaptive Following Methods.'

#### 4.10 Adaptive Following Methods (Kinect and Laser)

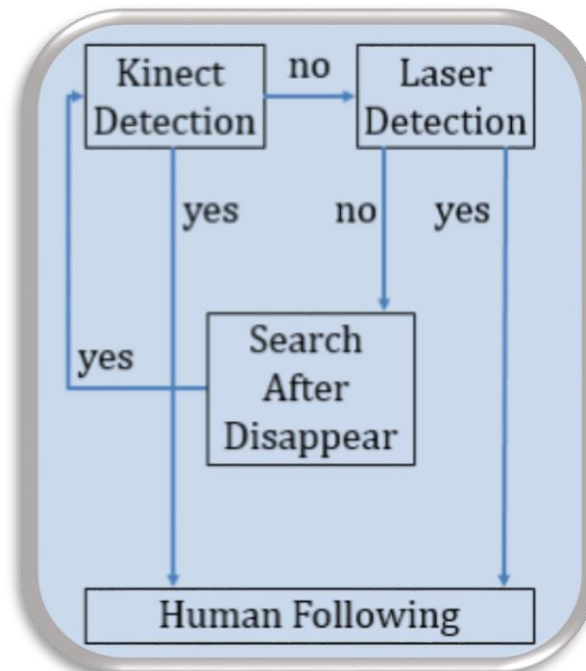
This adaptive method refers to *DF* and *HF* as an extension to the methods that are described above (*Direct-Following* Section 4.8 and *History-Following* Section 4.9). To use two sources of person detection, the method must implement ***a decision-making routine to select***

*which source to use for the following algorithm.* An additional parameter was created to compare the person's position determined by the laser detector in the world coordinates with that determined by the Kinect. If the distance between these two measurements is  $>20$  cm, then the method declares a "Match," which implies the same person is detected from both sources. Each sensor has inherent advantages and disadvantages as indicated in **Table 4**:

*Table 4- Properties of Kinect vs. laser detector*

	Kinect V2	Laser
<b>Horizontal FOV (degrees)</b>	84.1 RGB, 70.6 Depth	240
<b>Vertical FOV (degrees)</b>	53.8 RGB, 60 Depth	Only 2D view (20 cm high)
<b>Distance (meters)</b>	1-10 RGB, 0.5-4.5 Depth	0-30
<b>Reliability</b>	Depends (confidence level)	Less reliable

Based on empirical investigations, the decision-making routine gives priority to the Kinect (**Figure 20**) to calculate the following parameters. If after 3 s of "no" Kinect detection, the routine changes to the laser detector to calculate the following parameters until the Kinect recovers. If both sensors do not detect the person, then the routine uses the *SAD* algorithm.



*Figure 20 - Flowchart of the adaptive decision making Kinect- and laser-detection algorithm*

## 5. Chapter Five: Results and Discussion

### 5.1 Overview

The results of all the experiments are detailed in this chapter. The two preliminary experiments with Kinect V1 are described in Sections 5.2 and 5.3. The experiment that compares the different following angles used with Kinect V2 is described in Section 5.4. The comparison between the three occlusion detection algorithms is described in Section 5.5. The direct-following and history-following experiments that evaluate and compare the performances of various combinations of integrated algorithms in the two main human-following methods are described in Sections 5.6 and 5.7. The adaptive Kinect-laser vs non-adaptive (for direct following and history following) experiment is described in Section 5.8.

### 5.2 Preliminary Experiment: Identifying Testable Parameters on a No-Pan Kinect V1

A series of parameters were tested to evaluate the Kinect parameters (**Table 5**).

*Table 5- The set of usable parameters tested for the variables*

Max speed (m/s)	Responsiveness distance	Responsiveness turn	Minimum distance to person (m)	Angle (rad)	Results/Opinion
1	1.2	1.2	2	0.5	Loses the person (due to large angle)
1	1.2	1.2	2	0.4	Loses the person (due to large angle)
1	1.2	1.2	2	0.3	Loses the person (due to high velocity)
1	0.8	1.2	2	0.3	Causes the robot to vibrate and loses the person (due to high velocity)
1	0.8	0.8	2	0.3	Causes the robot to vibrate and loses the person (due to high velocity)
0.5	0.5	0.5	2	0.3	Slow near the person; some loss of the person (due to high velocity and far distance)
0.3	0.8	0.8	2	0.3	Good tracking but far distance from the person
0.3	0.8	0.8	1	0.3	Good tracking and good distance from the person

The robot lost the person in the first and second trials due to the large tracking angle (0.5 and 0.4 rad, respectively). In the third trial, the robot lost the person due to its high linear velocity (responsiveness of 1.2). After the responsiveness of the linear velocity (which depends on the distance between the robot and the person) was decreased to 0.8, the robot lost the person due to vibration caused by the angular responsiveness of 1.2. If the

responsiveness is too slow (0.5 for both linear and angular velocities) and the maximum speed of the robot is also slow (0.5 m/s), the robot loses the person due to the far distance. After obtaining good stable tracking at 0.3 rad from the side, with responsiveness of 0.8 for both linear and angular velocities and a maximum speed of 0.3 m/s, the minimum distance from the person can be decreased to 1 m instead of 2 m.

Based on the above experiments, the final selected parameters were:

- 0.3 rad for angle following
- 1 m for minimum distance between a person and the Kinect camera
- 0.8 responsiveness both for turning and distance
- 0.3 m/s as maximum speed of the robot

### 5.3 Preliminary experiment: Testing Objective & Subjective Metric Performances of Two Angles on a No-Pan Kinect V1

The following objective results were obtained:

- If the robot lost the person, it continued its current movement until it either found the person and then continued to follow the person or it detected an obstacle and stopped. This occurred in both ways.
- The maximum distance before losing the person was 4 m (see **Figure 21** for an example of losing the tracking at distances >4 m).
- Since the robot lacks a vertical tilt, it tended to lose track of tall people when they were too close to it (see **Figure 21** for an example of losing tracking at close distance).
- The robot detected shadows on the wall as additional people due to the lighting conditions and to reflections. This happened mostly at side-following because of the small distance between the robot and the wall (**Figure 22**).
- In 93% of instances of losing the person, the robot recovered by itself (26/28) (**Figure 22**).
- The average following angle was a bit larger than programmed (**Figure 23, Table 6**):
  - Back-following  $0.76^\circ$  (supposed to be  $0^\circ$ )
  - Side-following  $18.25^\circ$  (supposed to be  $17.19^\circ$ )

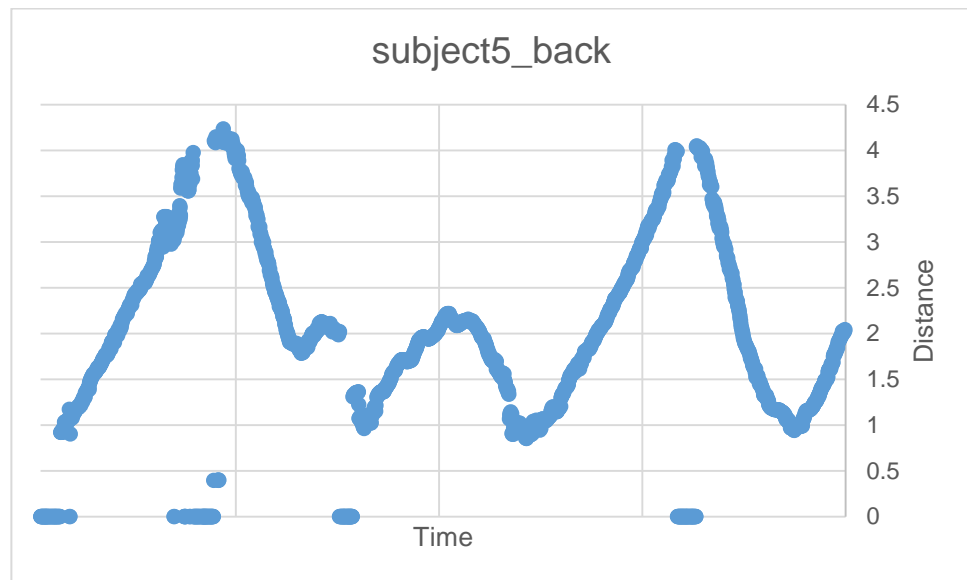


Figure 21 - Examples of losing tracking at distances  $> 4$  m and at close distances (Kinect V1)

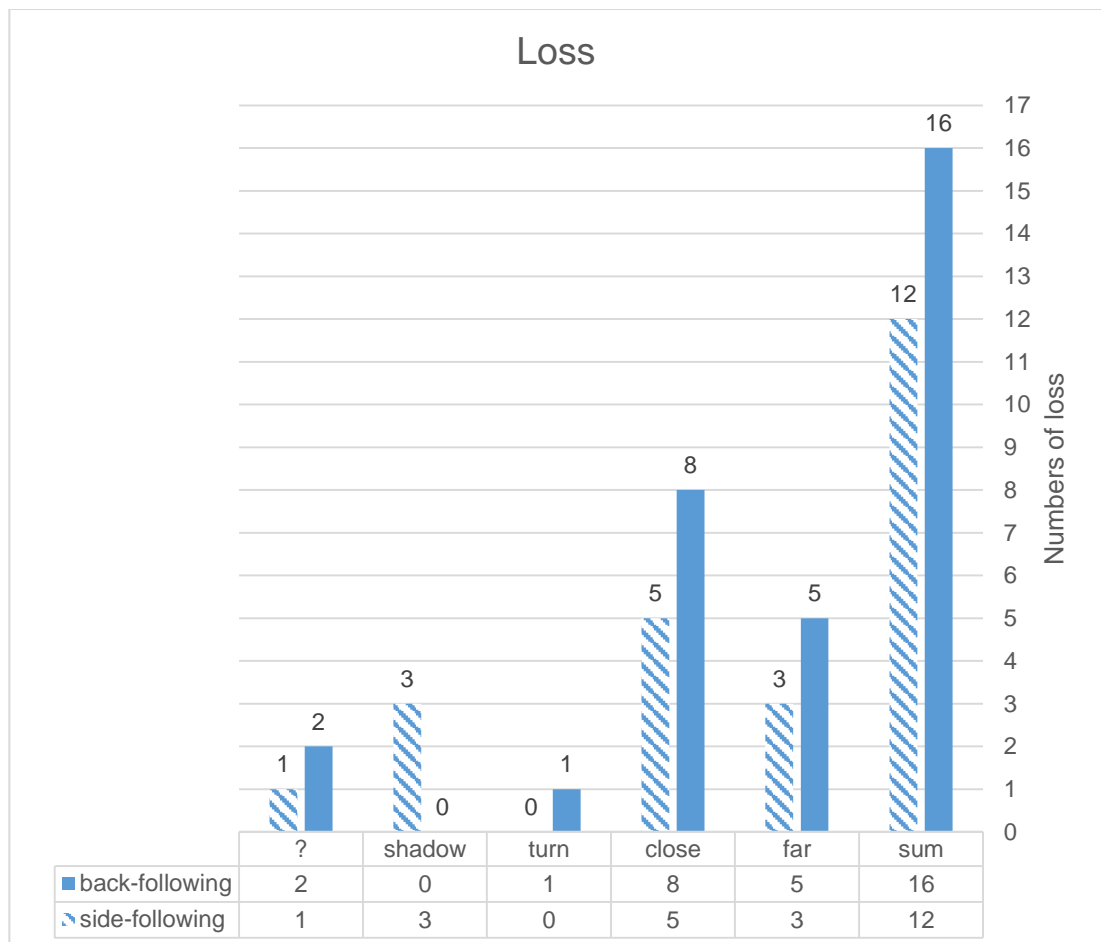


Figure 22 - Losses reasons (back-following vs. side-following)

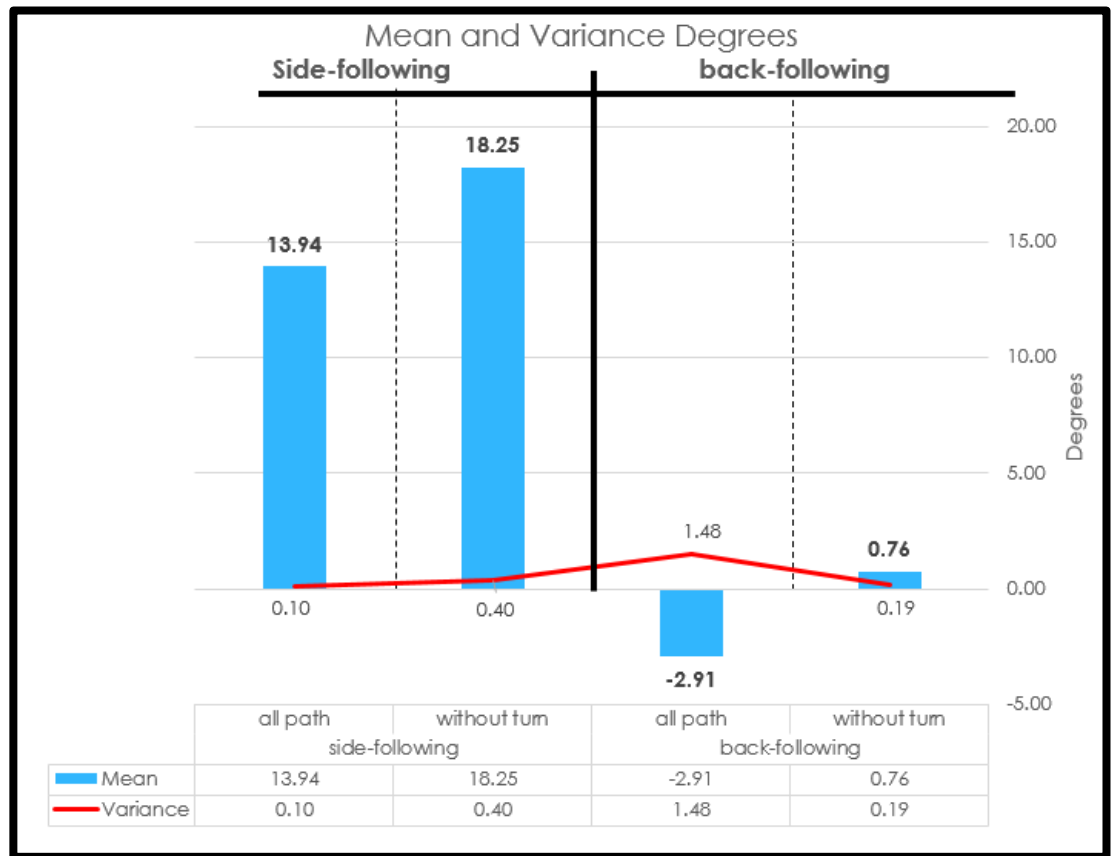


Figure 23 - Means and variance degrees (back-following vs. side-following)

Table 6 - Following angle results (back-following vs. side-following)

subject	side-following All path	side-following without turn	back-following All path	back-following Without turn
1	14.21	18.59	-3.4	0.93
2	13.72	17.96	-3.93	0.03
3	13.64	17.66	-1.25	0.46
4	13.72	17.61	-3.46	0.9
5	14.42	19.25	-1.48	0.98
6	13.94	18.42	-3.91	1.23
average	13.94	18.25	-2.91	0.76
covariance	0.10	0.40	1.48	0.19



The subjective results of the experiments are summarized in **Table 7**. The main findings were:

- There were significant differences in the subjective assessments of 'following quality', 'robot responsiveness' and 'comfort with speed.'
- The robot lost the person more frequently in back following (by 33% more losses in back-following) than in side-following. This is because people tended to walk faster (and noticed the robot's presence less) and created a much larger distance between them and the robot.
- Subjects felt that the robot moved too slowly and they thus lowered their speed to adapt to the robot; yet to 4/6 subjects the robot felt slower in back-following than side-following.
- The subjects reported that they adapted their walking speed and behavior to the robot to a greater extent for back-following.
- 4/6 of subjects felt less threatened with the following distance of the robot in back-following.
- Subjects were slightly less stressed by the robot in the back-following condition.
- The task was perceived as non-stressful. The robot was perceived to be friendly and not dangerous, scary, annoying or stressful.
- There was no conclusive self-reported preference between back-following and side-following.
- There were no perceived differences in the quality of following.
- The comfort of the subjects with the responsiveness of the robot was the same for both back and side-following.

Table 7- Subjective results of the performances of two angles experiment

Following behind							Average
Gender	F	M	F	M	M	F	
Subject #	1	2	3	4	5	6	
Person stressed by task	1	2	1	1	1	3	1.5
Person stressed by robot	2	2	1	1	1	1	1.33
Person adapted behavior based on robot	3	4	5	3	3	4	3.66
Robot adapted behavior based on person	4	3	3	4	4	4	3.66
Walking was independent of robot	1	2	1	2	2	2	1.66
Person was comfortable with speed of robot	1	4	3	3	2	5	3
Robot moved too slowly	5	4	5	4	5	1	4
Person was satisfied with the quality of following	2	4	5	3	3	5	3.66
Person felt safe regarding the distance of the robot	5	5	5	4	5	5	4.83
Person lowered speed to adapt to the speed of the robot	5	5	5	4	5	4	4.66
Following at angle							Average
Gender	F	M	F	M	M	F	
Subject #	1	2	3	4	5	6	
Person stressed by task	1	1	1	2	1	2	1.33
Person stressed by robot	2	3	1	2	1	3	2
Person adapted behavior based on robot	2	2	1	4	4	4	2.83
Robot adapted behavior based on person	4	3	5	4	4	4	4
Walking was independent of robot	2	3	2	2	2	1	2
Person was comfortable with speed of robot	3	4	5	3	2	5	3.66
Robot moved too slowly	4	3	1	4	4	2	3
Person was satisfied with the quality of following	4	4	5	3	2	4	3.66
Person felt safe regarding the distance of the robot	3	3	5	4	4	3	3.66
Person lowered speed to adapt to the speed of the robot	5	2	5	4	5	5	4.33
Subject #	1	2	3	4	5	6	
Person felt a difference in the difference between the 2 trials	4	4	3	2	4	4	3.5
Opinion of Robot (the robot is..)							
Friendly	3	3	3	2	5	4	3.33
Disturbing	2	3	1	4	3	1	2.33
Considerate	1	4	3	3	1	4	2.66
Dangerous	2	1	1	2	1	1	1.33
Scary	1	2	1	2	1	2	1.5
Annoying	3	2	1	3	3	1	2.16
Stressful	3	1	1	2	2	2	1.83

#### 5.4 Testing Objective & Subjective Metric Performances of Three Angles on a Pan Kinect V2

Objective indicators used to assess the quality of following (Table 8) indicated while that the mean following angle at 0° following and 30° following were consistently close to the intended angles (2.31° and 28.26°, respectively), the implementation of 60° following was unsuccessful (mean = 26.11°, STD = 7.315). This lack of success may be attributed to the different methods of following and problems that were caused due to the implementation of

the 60° of following. **There was no significant difference across trials in the following distance (0° and 30°)** (Honig et al., 2016). The mean following distance for men was 17.2 cm greater (by 8%) than that for women. The number of times the robot lost track of the participants (defined as 'losses' in **Table 8**), and the mean number of interventions due to loss or for safety reasons increased with the following angle, but these differences were not statistically significant (sig. = 0.206, sig. = 0.205, sig. = 0.297, respectively).

*Table 8- Cumulative results for objective measures for quality of walk*

	0°	30°	60°
<b>Distance</b>	2.36±0.69	2.29±0.59	2.2±0.57
<b>Angle</b>	2.31±10.41	-28.26±11.04	26.11±12.73
<b>Number of losses</b>	0.36±0.75	0.72±0.62	1.16±0.28
<b>Number of interventions due to loss</b>	0.32±0.62	0.44±0.82	0.96±0.97
<b>Number of interventions due to safety</b>	0.08±0.27	0.24±0.43	0.52±0.71

## 5.5 Comparison of Occlusion Algorithms

The results (**Table 9**) show that the algorithm that uses depth information (**DO**) yielded better average true detection ( $92.7\% \pm 8.75\%$ ) than the algorithms that use grey level images (**VO**) ( $40.5\% \pm 23\%$ ) and their combination (**CO**) ( $63\% \pm 11.25\%$ ) in corrent implementation for all scenarios and distances, including very far distances (8 m). **DO** exhibited high sensitivity and specificity as compared to **VO** (sig. = 0.000) and **CO** (sig. = 0.000). **DO** yielded the best detection results, followed by the **CO** algorithm, and the worst algorithm was **VO** (sig. = 0.000,  $F = 25.748$ ). **DO** yielded the least false alarms and misses ( $9.7\% \pm 8.75\%$ ), less than **CO** ( $36.9\% \pm 11.25\%$ ) but not statistically significantly different, and the worst results were obtained for **VO** ( $59.4\% \pm 23\%$ ) (sig. = 0.173,  $F = 1.98$ ). These results indicate the reliable measurements of the Kinect Depth stream that are used in the **DO** algorithm developed to derive a compatible threshold to reduce false distance measurements and closer body parts mistakes. The low performance of **VO**, namely, the high number of false occlusions detected showed that the detection of straight lines was not adequate.

Table 9- results of comparison occlusion algorithms experiment

	True Negative	False Positive	False Negative	True Positive
<b>DO (occlusions)</b>	0.88±0.17	0.12±0.17	0.083±0.09	0.916±0.09
<b>CO (occlusions)</b>	0.895±0.12	0.105±0.12	0.552±0.06	0.447±0.06
<b>DO (wall)</b>	1±0	0±0	0.088±0.09	0.912±0.09
<b>VO (wall)</b>	0.695±0.31	0.305±0.31	0.883±0.15	0.116±0.15
<b>CO (wall)</b>	0.94±0.08	0.06±0.08	0.76±0.19	0.24±0.19

\*The best results are highlighted in yellow

The results indicate that **CO** detection was almost half of **DO** detection at distances <5 m (Pearson: sig. = 0.01, r (43) = 0.706) and almost half of **VO** at distances >5 m (Pearson: sig. = 0.01, r (22) = 0.535). **VO** produced a high number of false negatives. Unexpectedly, **DO** was found to be better than **VO** and **CO** even at far distances.

## 5.6 Direct-Following Experiment

The main results obtained were:

- The results presented in **Table 10** indicate that the trials with the two search algorithms (trials 1 and 2) gave better results than the trial without the search algorithm (trial 3) in terms of the percentage of self-recoveries and intervene-recovers out of the total number of losses (with a search algorithm, all losses were self-recovered; without the search algorithm, all the losses required intervention for 100% recovery). The only significant difference between the three trials was in the ratio of stable tracking of the person to no tracking of the person (sig. = 0.047, f = 3.633). The results for average ratio of stable tracking of the person to no tracking of the person for the trials with the two search algorithms were better by 21% (trials 1 and 2, mean±STD = 0.983±0.016) than for the trial with no search algorithm (trial 3, mean±STD = 0.81±0.165). Post-hoc pairwise comparisons (Tukey test) showed a difference that was almost significant between trial 3 and trials 1 and 2 (sig. = 0.075, p = 0.078, respectively), with a homogeneity of variances of 0.11.
- For the seven subjects, there were differences in the total number of losses: trials with the search algorithms yielded 60% less losses than those without a search algorithm (search 0.4±0.489; without search 1±1.264) but the homogeneity of variances of 0.01 indicates that this difference was not significant. It was also found, as expected, that each subject walked at a unique average velocity.

- There was no significant difference between the two search algorithms. The only difference between trial 1 and trial 2 was that trial 1 was conducted with the **DO** algorithm, and trial 2, without.
- The order in which the trials were conducted during the experiment did not influence the results.
- In this experiment the best algorithm was the combination of **SAD** and **DO** (trial 2 - **Figure 24**) because the combination gave significantly better results – by 21% – than the **DO** algorithm alone (without the search algorithm) for average ratio of stable tracking of the person to no tracking of the person; this result was similar to that of the first trial, which used only the search algorithm without the **DO** algorithm (**Table 10**).

*Table 10 - Results for Direct-Following experiment*

Trial	Type	Total loss	Total loss with self recovery	Ratio self-recovery to total loss	Total loss with intervent	Ratio intervent to total loss	Total safety intervent	Robot distance
1	Search	2	2	100	0	0	3	13.28
2	Search+ occlusion	4	4	100	0	0	1	12.74
3	Occlusion	8	0	0	8	100	3	13.85

Trial	Velocity of the subject	Depth occlusion	False alarm depth occlusion	Ratio track to no-track	Distance	STD distance	Total obstacles hit
1	0.54			0.98	3.14	0.83	2
2	0.56	0.84	0.46	0.98	2.99	0.71	2
3	0.52	0.79	0.49	0.81	3.29	0.86	1

\*Yellow fill indicates the best results (although not statistically significant)

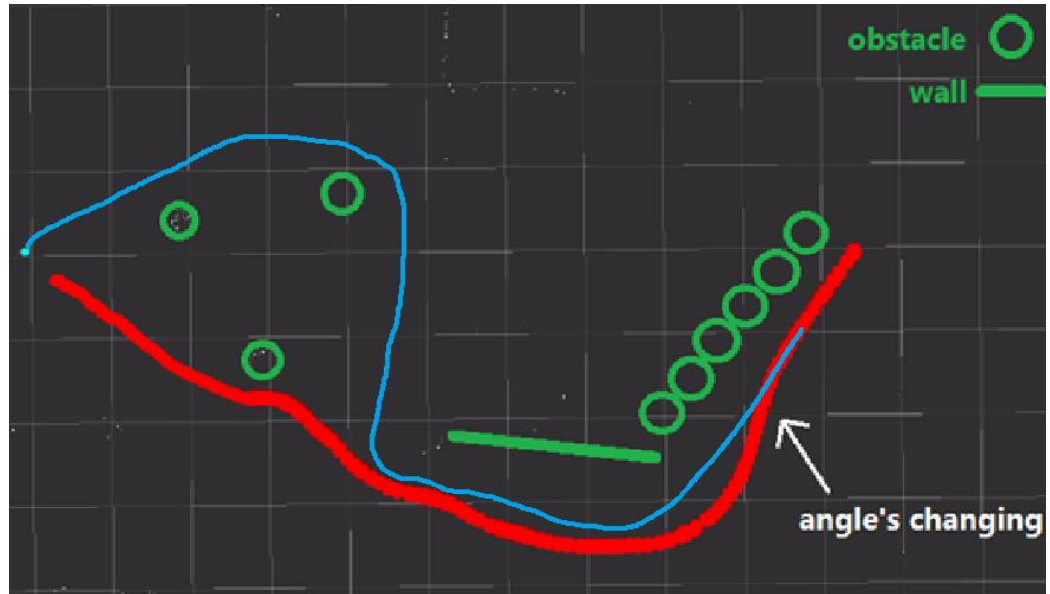


Figure 24- An example of Direct-Following with search and occlusion algorithms from RVIZ (Blue-subject, Red-robot)

### 5.7 History Following Experiment

The results in **Table 11** indicate that there was only one significant difference between the two *HF* algorithms, namely, the difference in the total path length of the robot during the experiment (sig. = 0.02,  $f = 15.765$ ). The algorithm that gave the longer path – by 22% – was the *DO* algorithm used in the second trial (without *DO*  $16.085 \pm 1.604$ ; with *DO*  $19.157 \pm 1.007$ ). In addition, there was no significant difference in the number of losses or in the ratio of stable tracking to no tracking of the person. There was a difference in the percent of self-recoveries—200% more self-recoveries with the *DO* algorithm (7.14% self-recoveries for no *DO* algorithm vs 21.43% self-recoveries with the *DO* algorithm).

As in the *DF* experiment (Section 5.6), each subject had a unique average velocity, as expected, and the order of the trials during the experiment did not influence the results.

A reasonable explanation for the difference in the length of the path is that this parameter depends on the movements of the robot during recognizing an occlusion of the person (by *DO* algorithm) and changing the following angle to change the line of sight between the robot and the person (which increases the length of the path). In addition, the percent of self-recoveries after loss was three times higher when using the *DO* algorithm; this finding implies that it is better to use the *DO* algorithm in an unknown environment. Hence, the best algorithm in this experiment was shown to be the *DO* algorithm (**Table 11** and **Figure 25**).

Table 11- Results for History-Following experiment

Trial	Type	Total loss	Total loss with self recovery	Ratio self-recovery to total loss	Total loss with intervent	Ratio intervent to total loss	Total safety intervent	Robot distance
1	Search	8	1	12.5	7	87.5	2	16.08
2	Search+ occlusion	9	3	33.3	6	66.6	4	19.16

Trial	Velocity of the subject	Depth occlusion	False alarm depth occlusion	Ratio track to no-track	Distance	STD distance	Total obstacles hit
1	0.50			0.81	3.29	0.92	1
2	0.48	0.80	0.51	0.82	3.50	0.91	1

\*yellow fill indicates statistically significant result.



Figure 25 - Example of History-Following with search and occlusion algorithms from RVIZ (Blue-subject, Red-robot)

A comparison of the three trials of the *DF* experiment (Section 5.5) and the two trials of the *HF* experiment (**Table 12**) shows that direct following is better than history following (the opposite than expected) due to large calculation processing in *HF*. Losses for *DF* were 45% less than those for *HF* (sig. = 0.077, not significant). Out of the total losses, the percentage of intervene-recoveries was lower by 70% in the *DF* (sig. = 0.003). The tracking ratio is 13%

better in the *DF* (sig. = 0.022) and the path distance of the robot was shorter by 25% (sig. = 0.000). In addition, the velocity of the subject was faster by 10% in the *DF* (sig. = 0.071, not significant).

*Table 12- Statistic comparison between 3 DF trials vs 2 HF trials*

Variable	3 Direct trials Mean±STD	2 History trials Mean±STD	F	Sig.
Number of total losses	0.67±0.856	1.21±0.893	3.332	0.077
Percent of intervene-recover out of total loss	0.1905±0.402	0.6429±0.412	10.404	0.003
Tracking ratio (Kinect)	0.9257±0.129	0.8157±0.136	5.804	0.022
Average velocity of the subject (m/s)	0.5419±0.081	0.4877±0.083	3.484	0.071
Total path length of the robot (m)	13.29±1.989	17.6214±2.114	37.868	0.000

\*Yellow fill indicates that the difference is significant at the 0.05 level.

The results indicate influence of the different following methods (*DF* vs *HF*) on the results (**Table 13**). The next experiment must include a larger number of participants in order to obtain statistically significant results.

*Table 13 - Results for the comparison between 3 DF trials vs 2 HF trials*

Trial	Type	Total loss	Total loss with self recovery	Ratio self-recovery to total loss	Total loss with intervent	Ratio intervent to total loss	Total safety intervent	Robot distance
1	Direct	4.67	2	66.66	2.67	33.33	2.33	13.29
2	History	8.50	2	22.90	6.50	77.05	3	17.62

Trial	Velocity of the subject	Depth occlusion	False alarm depth occlusion	Ratio track to no-track	Distance	STD distance	Total obstacles hit
1	0.54	0.81	0.47	0.93	3.14	0.80	1.67
2	0.49	0.80	0.51	0.82	3.39	0.92	1

\*Green indicates the best results (although not significant), and yellow indicates results with statistical significance at the 0.05 level.

The main significant difference between the five trials lies in the total path length of the robot (**Figure 26** and **Table 14**), with homogeneity of variances of 0.74 (sig. = 0.000, f =



14.614). The shortest path (trial 2- *DF* with *SAD* and *DO*) was shorter by 33% than the longest path (trial 5- *HF* with *SAD* and *DO*).

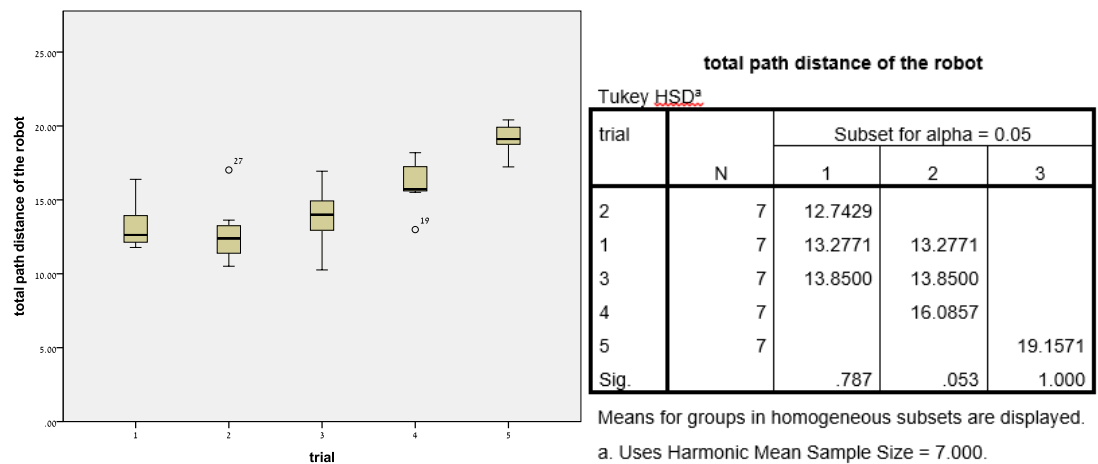


Figure 26 - Five trials total path length of the robot

Table 14 - Results for five trials

Trial	Type	Total loss	Total loss with self recovery	Ratio self-recovery to total loss	Total loss with intervent	Ratio intervent to total loss	Total safety intervent	Robot distance
1	Direct search	2	2	100	0	0	3	13.28
2	Direct search+ occlusion	4	4	100	0	0	1	12.74
3	Direct occlusion	8	0	0	8	100	3	13.85
4	History search	8	1	12.5	7	87.5	2	16.08
5	History search+ occlusion	9	3	33.3	6	66.6	4	19.16

Trial	Velocity of the subject	Depth occlusion	FA depth occlusion	Ratio track to no-track	Distance	STD distance	Total obstacles hit
1	0.54			0.98	3.14	0.83	2
2	0.56	0.84	0.46	0.98	2.99	0.71	2
3	0.52	0.79	0.49	0.81	3.29	0.86	1
4	0.50			0.81	3.29	0.92	1
5	0.48	0.80	0.51	0.82	3.50	0.91	1

## 5.8 Adaptive Kinect-Laser Method vs. Non-Adaptive Kinect Method (for Direct-Following and History-Following) Experiment

The comparison of the **adaptive Kinect-laser** methods with the **non-adaptive Kinect** methods indicates better performance of the adaptive methods (**Table 15**):

*Table 15 – Statistic comparison between Adaptive Kinect-laser methods vs Non-adaptive Kinect methods*

Variable	Adaptive Method Mean±STD	Non-Adaptive Method Mean±STD	F	Sig.
Number of total losses**	1.25±0.812 **0.982±0.54	2.27±1.18 **0.592±0.49	13.395	0.000
Percent of intervene-recover out of total losses	0.225±0.362	0.651±0.364	32.897	0.000
Number of safety interventions	0.79±0.713	1.33±0.781	12.588	0.001
STD of the distance between robot and subject (m)	0.785±0.261	0.905±0.306	4.266	0.042
Percent of false depth occlusions	0.237±0.114	0.296±0.147	4.866	0.03
Percent of depth occlusions	0.774±0.130	0.715±0.122	5.132	0.026

\*Yellow fill indicates that the difference is significant at the 0.05 level.

\*\*The total number of losses has a homogeneity of variances of 0.008. Therefore, a transformation of 1/X was performed, resulting in a homogeneity of variances of 0.561.

The percent of 'intervene-recovers' out of total losses was lower in the adaptive Kinect-laser methods by 65% (adaptive Kinect-laser average 0.225±0.362 vs. non-adaptive average 0.651±0.364) due to the ability of the robot to 'reconnect' with the person by using the laser sensor when the Kinect had lost the person. For the same reason, the number of interventions resulting from the robot getting too close to a wall was lower in the adaptive Kinect-laser methods by 41% (adaptive Kinect-laser average 0.79±0.713; non-adaptive average 1.33±0.781). In addition, the STD of the distance between the robot and the subject was smaller in the adaptive Kinect-laser methods by 13% (adaptive Kinect-laser average 0.785±0.261; non-adaptive average 0.905±0.306) due to fewer losses and more steady following with less variation in the distances. Perhaps, this is also the reason for better occlusion detection and less false occlusion detection in the adaptive Kinect-laser methods by 8% and 20%, respectively.

The comparison between the two following methods (*DF adaptive and non-adaptive methods* vs *HF adaptive and non-adaptive methods*) shows better results for the *DF* (**Table 16**):

*Table 16- Direct-Following (adaptive and non-adaptive methods) vs History-Following (adaptive and non-adaptive methods)*

Variable	Direct Following Mean±STD	History Following Mean±STD	F	Sig.
Number of total losses	1.35±1.101	2.17±1.018	14.097	0.000
Number of intervene-recover	0.77±1.057	1.38±1.064	7.789	0.006
Number of laser obstacles found	37.29±19.64	50.69±23.46	9.2	0.03
Percent false alarm detection of legs	0.189±0.302	0.435±0.295	7.98	0.007
Tracking ratio for Kinect	0.935±0.081	0.88±0.094	9.174	0.003
Percent of false depth occlusion	0.225±0.125	0.309±0.131	10.373	0.002
Velocity of the subject	0.404±0.092	0.336±0.079	14.82	0.000
Total path length of the robot	19.298±1.80	23.607±2.08	64.718	0.000
Number of matchings for the position of the subject by Kinect and laser detector	32.08±22.11	15.5±13.99	9.641	0.003
Tracking ratio for laser detector	0.596±0.264	0.318±0.186	17.689	0.000

\*Yellow fill indicates difference is significant at the 0.05 level.

Higher – by 6% and 47%, respectively – stable tracking ratios were obtained with the Kinect and laser sensors for the *DF* methods (adaptive and non-adaptive). A reasonable explanation for this finding is that the robot always turns directly to the subject in *DF* methods, which means that if the robot loses the person, the Pan returns the Kinect to the center of the robot where the person is most likely to be and the line of sight of the laser sensor is aimed in front of the robot. In contrast, in the *HF* (adaptive and non-adaptive) methods, the robot does not move directly to the person but rather to the person's historical position. In other words, most of the time the person is not in front of the robot. For the same reason, the number of interventions due to losses was lower, by 44%, in *DF* (adaptive and non-adaptive) methods than in *HF* (adaptive and non-adaptive) methods due to the direction of the sensors when the robot loses the person and to the inferior ability of the robot to self-recover when the sensors are not directed to the person (in front of the robot). This reasoning also explains the lower – by 38% – number of total losses for *DF* (adaptive and non-adaptive) methods. In addition, less stable tracking can cause more false occlusion detections due to false person detection.

The average distance between the robot and the subject was not significantly different (homogeneity of variances of 0.012;  $DF$  3.051 $\pm$ 0.545;  $HF$  3.696 $\pm$ 0.370 with sig. = 0.000,  $f = 45.905$ ). The percent of false alarms of legs detection was higher in  $HF$  (adaptive and non-adaptive) by 56% and showed a positive correlation to the distance. Longer distances could cause a higher probability of false alarms, since it becomes more difficult for the laser sensor to find the person's legs as the distance increases. The velocity of the subject could be faster when the robot is closer and slower when the robot is further away (since the person may have to wait for the robot). The number of matchings between the position of the subject by the Kinect and by the laser sensor was higher when both sensors give stable and reliable tracking, which also depends on the distance.

The total path length of the robot was shorter – by 18% – in  $DF$  (adaptive and non-adaptive), as expected. In  $DF$  (adaptive and non-adaptive), the robot moves directly to the subject by taking short cuts, while in  $HF$  (adaptive and non-adaptive) the robot moves to the historical position of the subject without any short cuts. The difference may explain why the number of laser obstacles in  $HF$  (adaptive and non-adaptive) was higher by 36%, since the robot follows the subject's historical position when the subject moves near an obstacle.

In addition, people preferred the  $DF$  than  $HF$  due to the robot's response (location and continuity of movement).

A comparison between the heights of the male and female participants showed that the men were significantly taller than the women by 4% and the standard deviation (STD) of the distance between the robot and the subject (**Table 17**) of the men were significantly higher than the women by 3%.

*Table 17- Statistical comparison between males vs females*

Variable	Males Mean $\pm$ STD	Females Mean $\pm$ STD	F	Sig.
Height of the subject	1.635 $\pm$ 0.626	1.593 $\pm$ 0.060	10.95	0.001
STD of the distance between robot and subject	0.792 $\pm$ 0.279	0.921 $\pm$ 0.289	4.836	0.03

\*Yellow indicates difference is significant at the 0.05 level.

There was no difference in the tracking ratio or the average distance between the robot and the subject that can explain the different STDs. The only reasonable explanation lies in the fluctuations in the velocity of the subject. The velocity calculation was based only the average velocity during the entire trial and not the fluctuation of the velocity during the trial. If males, for instance, walked at a "more" constant velocity than females, then the STD distance of males are smaller by 14% than females.

For the 24 participants, the only significant difference between the men and the women was that in their height, as expected (sig. = 0.000,  $f = 22.775$ ).

The order of the trials during the experiment did not influence on the results.

A comparison between the **four trials** (*HF-Adaptive*, *HF-Non-Adaptive*, *DF-Adaptive*, *DF-Non-Adaptive*) shows differences (**Table 18**). The means for groups in homogeneous subsets are displayed in

**Figure 27.**

*Table 18- Statistical comparison between the four trials*

Variable	<i>HF-Adaptive</i> Method Mean± STD	<i>HF-Non-Adaptive</i> Mean± STD	<i>DF-Adaptive</i> Mean± STD	<i>DF-Non-Adaptive</i> Mean± STD	F	Sig.
Number of total losses	1.67±0.761	2.67±1.007	0.83± 0.637	1.88±0.899	19.39	0.000
Number of safety interventions	0.88± 0.797	1.46±0.721	0.71±0.624	1.21±0.833	4.838	0.004
Average distance between robot and subject	3.614±0.39	3.778±0.33	2.985±0.50	3.118±0.58	16.21	0.000
Tracking ratio with Kinect	0.872±0.11	0.888±0.06	0.950±0.05	0.919±0.09	3.664	0.015
Percent of false depth occlusion	0.285±0.10	0.333±0.15	0.189±0.10	0.260±0.13	5.43	0.002
Percent of depth occlusions detected	0.729±0.15	0.750±0.06	0.818±0.07	0.681±0.15	5.360	0.002
Velocity of the subject	0.327±0.07	0.346±0.08	0.404±0.09	0.404±0.09	5.051	0.003
Total path length of the robot	23.40±2.10	23.80±2.08	18.60±1.72	19.99±1.62	43.52	0.000

\*Yellow highlighting indicates the best results and black highlighting, the worst, with difference being significant at the 0.05 level.

Total Loss					Safety Intervention				Average Distance			
Tukey HSD <sup>al</sup>					Tukey HSD <sup>a</sup>				Tukey HSD <sup>a</sup>			
trial	N	Subset for alpha = 0.05			trial	N	Subset for alpha = 0.05		trial	N	Subset for alpha = 0.05	
		1	2	3			1	2			1	2
3	24	.8333			3	24	.71		3	24	2.9850	
1	24		1.6667		1	24	.88		4	24	3.1183	
4	24		1.8750		4	24	1.21	1.21	1	24		3.6142
2	24			2.6667	2	24		1.46	2	24		3.7788
Sig.		1.000	.825	1.000	Sig.		.102	.655	Sig.		.754	.612

Ratio Track Kinect				False Depth Occlusion				Depth Occlusion			
Tukey HSD <sup>a</sup>				Tukey HSD <sup>a</sup>				Tukey HSD <sup>a</sup>			
trial	N	Subset for alpha = 0.05		trial	N	Subset for alpha = 0.05		trial	N	Subset for alpha = 0.05	
		1	2			1	2			1	2
1	24	.87221		3	24	.1896		4	24	.6812	
2	24	.88842	.88842	4	24	.2604	.2604	1	24	.7292	.7292
4	24	.91958	.91958	1	24		.2854	2	24	.7500	.7500
3	24		.95000	2	24		.3333	3	24		.8188
Sig.		.252	.081	Sig.		.216	.194	Sig.		.206	.056

Subject's Velocity				Robot's Distance			
Tukey HSD <sup>a</sup>				Tukey HSD <sup>a</sup>			
trial	N	Subset for alpha = 0.05		trial	N	Subset for alpha = 0.05	
		1	2			1	2
1	24	.3271		3	24	18.6055	
2	24	.3458	.3458	4	24	19.9912	
3	24		.4042	1	24		23.4055
4	24		.4042	2	24		23.8097
Sig.		.877	.099	Sig.		.062	.882

Figure 27-Means for groups in homogeneous subsets

The Kinect tracking ratio was highest in the *DF*-adaptive Kinect-laser mode as compared to the ratio for the other modes (*DF*-non-adaptive by 3%; *HF*-adaptive by 9%; *HF*-non-adaptive by 7%). There was a significant difference only between the best and the worst results (*DF*-adaptive  $0.95 \pm 0.05$  vs *HF*-adaptive  $0.872 \pm 0.11$ , with sig. = 0.015). A reasonable explanation for this finding is that the robot always turns directly to the subject in *DF* (adaptive and non-adaptive) methods, which means that if it loses the person, the Pan returns the Kinect to the center of the robot where the person is most likely to be. In contrast, in the *HF* (adaptive and non-adaptive) methods the robot does not move directly to the person but rather to the person's historical position. In other words, most of the time, the person is not in front of the robot. The results of false occlusion detection separates the *DF*-adaptive (once again the best result) to the *HF*-adaptive and the *HF*-non-adaptive with sig. = 0.002. Surprisingly, detection of real occlusions did not correlate with the false occlusions detections. The only significant difference is between *DF*-adaptive with the best result

( $0.818 \pm 0.07$ ) that is higher by 20% than the *DF*-non-adaptive ( $0.681 \pm 0.15$ , with sig. = 0.001).

The average distance between the robot and the subject in both *DF* methods (*DF*-adaptive  $2.985 \pm 0.5$ ; *DF*-non-adaptive  $3.118 \pm 0.58$ ) was shorter by 17% than that in both the *HF* methods (*HF*-adaptive  $3.614 \pm 0.39$ ; *HF*-non-adaptive  $3.778 \pm 0.33$ , with sig. = 0.000). The explanation of this significant difference lies in the differences between *DF* and *HF*, as explained above. This distance difference can also explain the velocity of the subject with significant differences between the *DF* (adaptive and non-adaptive) methods that are faster by 20% than the *HF*-adaptive. The velocity of the subject can be faster when the robot is closer and slower when the robot is farther (if waits for the robot).

There was a significant difference in the average number of interventions due to safety reasons between the *DF* and *HF* methods, with adaptive Kinect-laser methods of following having 46% less safety interventions (*DF*-adaptive  $0.71 \pm 0.624$ ; *HF*-adaptive  $0.88 \pm 0.797$ ) than the *HF*-non-adaptive mode ( $1.46 \pm 0.721$ ), with sig. = 0.004 and sig. = 0.04, respectively. The reason for this difference lies in the stability of the tracking, i.e., the adaptive-following methods use two sensors, and in the smaller number of losses – by 72% – in the adaptive-following methods than in the non-adaptive-following methods. The average number of losses was significantly different between trials. The *DF*-adaptive mode gave fewer losses ( $0.833 \pm 0.637$ ) than the *HF*-adaptive ( $1.666 \pm 0.761$ ), the *DF*-non-adaptive ( $1.875 \pm 0.899$ ) and the *HF*-non-adaptive ( $2.666 \pm 1.007$ ) by 50% (sig. = 0.005), 56% (sig. = 0.000) and 69% (sig. = 0.000), respectively.

The total path length of the robot was shorter by 18% in the *DF* (adaptive and non-adaptive) trials than in the *HF* (adaptive and non-adaptive) trials, as expected. There was a significant difference between the *DF* methods (*DF*-adaptive  $18.605 \pm 1.72$ ; *DF*-non-adaptive  $19.991 \pm 1.62$ ) and the *HF* methods (*HF*-adaptive  $23.405 \pm 2.1$ ; *HF*-non-adaptive  $23.809 \pm 2.08$ ), with sig. = 0.000. In the *DF* (adaptive and non-adaptive) methods, the robot moves directly to the subject with taking short cuts, as necessary, while in *HF* (adaptive and non-adaptive) methods the robot moves to the historical position of the subject without taking short cuts. Examples of trials are shown in **Figure 28**, where

1. *DF*-adaptive - loss near the third obstacle; robot maintains stable tracking around the corner; Markers of Kinect and laser position
2. *DF*-adaptive - "short-cut" at the third obstacle; loss of tracking around the corner and self-recovery; Markers of Kinect and laser position
3. *DF*-non-adaptive - "short-cut" at the third obstacle; stable tracking, even around the corner

4. *DF*-non-adaptive - "short-cut" at the third obstacle; loss around the corner and self-recovery
5. *HF*-adaptive - stable tracking even around the corner; Markers of Kinect and laser position
6. *HF*-adaptive - two losses near the walls, another loss around the corner and self-recovery; Markers of Kinect and laser position
7. *HF*-non-adaptive - two losses near the third obstacle; robot maintained stable tracking around the corner
8. *HF*-non-adaptive - "short-cut" at the third obstacle and loss of tracking, some losses near the walls and around the corner with self-recovery



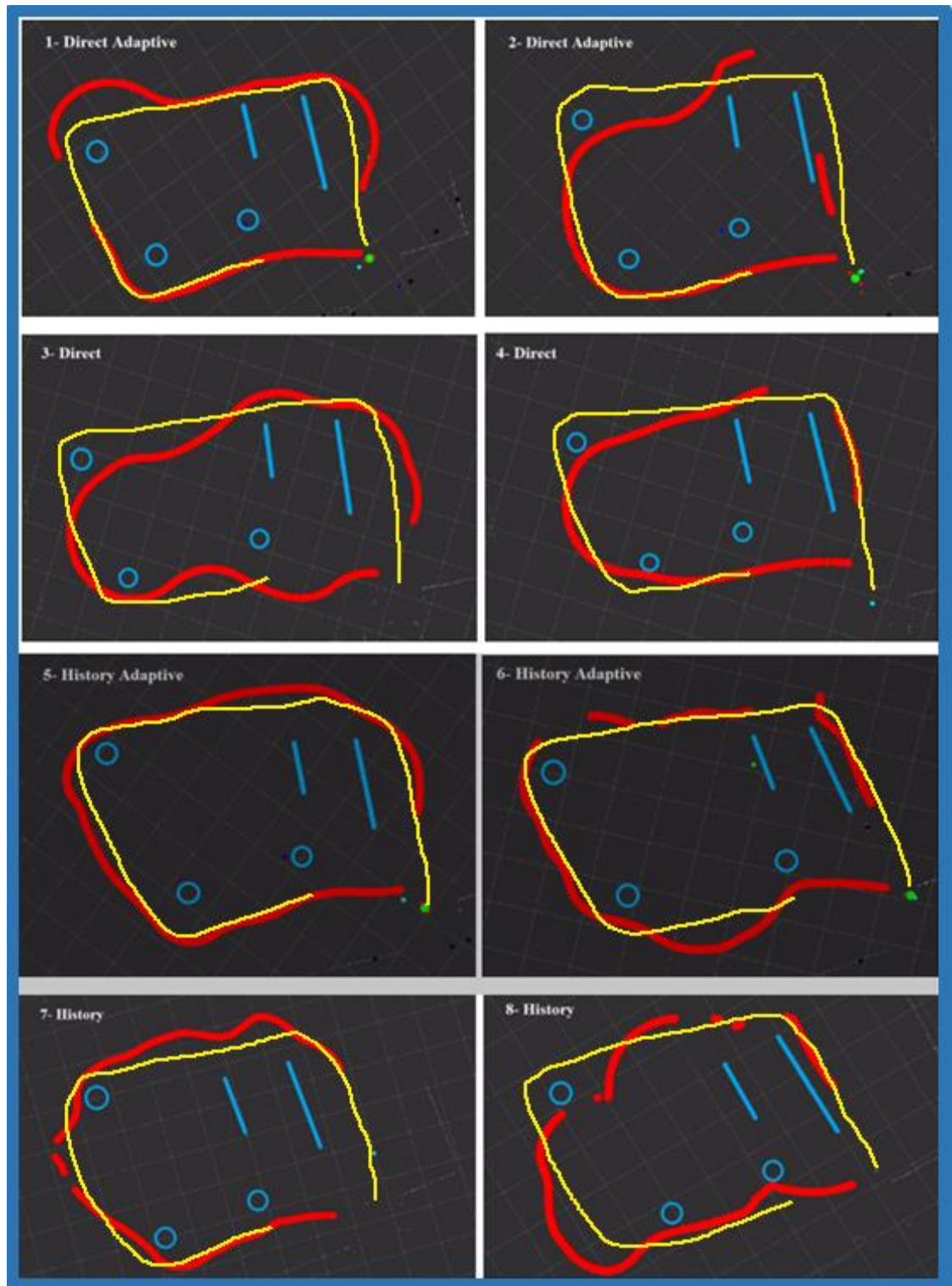







Figure 28- Trials examples for the adaptive and non-adaptive experiment (Yellow-subject, Red-robot)

-  -Green point with 2 small red points represent the position of the person by laser sensor
-  -Small turquoise point represents the position of the person by Kinect
-  -Straight blue line represents a wall
-  -Blue circle represents an obstacle
-  -Red line represents a stable robot following path

The summarized results for all the performance measures that were tested in the experiments are shown in **Table 19**. According to the results, the best trial was *DF*-adaptive (the best in all performance measures), followed by *HF*-adaptive, *DF*-non-adaptive and *HF*-non-adaptive, in that order.

*Table 19- The summarized results of the adaptive and non-adaptive (DF and HF) experiment ranking*

	History Adaptive	History	Direct Adaptive	Direct
<b>Total Loss</b>	1.67	<b>2.67</b>	<b>0.83</b>	1.88
<b>Total loss with self- recovery</b>	0.88	0.71	0.71	0.46
<b>Ratio self- recovery to total loss</b>	52.50	26.56	85.00	24.44
<b>Total loss with intervent</b>	0.79	1.96	0.13	1.42
<b>Ratio intervent to total loss</b>	47.50	73.44	15.00	75.56
<b>Total safety intervent</b>	<b>0.88</b>	<b>1.46</b>	<b>0.71</b>	1.21
<b>Obstacles hit</b>	0.17	0.33	0.00	0.38
<b>Laser obstacles</b>	50.42	50.96	39.38	35.21
<b>Match</b>	15.50		32.08	
<b>Total Kinect fall</b>	2	3	3	2
<b>Robot distance</b>	<b>23.45</b>	<b>24.18</b>	<b>18.69</b>	<b>20.16</b>
<b>Velocity of the subject</b>	0.33	0.35	0.40	0.40
<b>Depth occlusion</b>	0.73	0.75	<b>0.82</b>	<b>0.68</b>
<b>False alarm depth occlusion</b>	<b>0.29</b>	<b>0.33</b>	<b>0.19</b>	0.26
<b>Ratio track to no-track Kinect</b>	<b>0.87</b>	0.89	<b>0.95</b>	0.92
<b>Ratio track to no-track Laser</b>	0.32		0.60	
<b>Legs false alarm</b>	0.44		0.19	
<b>Distance between robot to subject</b>	<b>3.61</b>	<b>3.78</b>	<b>2.99</b>	<b>3.12</b>
<b>STD distance</b>	0.90	0.97	0.67	0.84

\*Yellow fill indicates the best results and black fill, the worst, with significant difference at the 0.05 level.

## 6. Chapter Six: Conclusions and Future Work

### 6.1 Conclusions

In this thesis, different algorithms and following methods were developed and tested for a human-following robot operating in unknown environments. The algorithms were developed for a robot operating without any a-priori information about the environment and without any special carry-on item and for people not wearing any specific items of clothing. The aim was to reduce the number of robot's losses of the person being detected and improve the robot's ability to self recover in unknown environments. The algorithms were implemented on a Pioneer LXRobot mobile platform equipped with a Kinect and laser sensor.

The algorithms use depth methods to improve the occlusion detection process. It uses the laser to avoid obstacles during the following process in real time, adapts to the linear and angular velocities of the robot and it remembers the last position of the person to search the person after disappear by moving to the person last position and turns to the direction, that was calculated.

The main conclusions from this research were:

- The best occlusions detection algorithm is the *DO*, which uses the depth information from the Kinect.
- For both following methods (*DF* and *HF*), the best performance was achieved by integrating three algorithms—*DO*, *OA*, and *SAD*.
- Adaptive methods that combine the laser sensor with the Kinect for *DF* and *HF* methods are better than the methods that do not use the laser (non-adaptive), and *DF* methods are better than *HF* methods. The final ranking of algorithms is: *DF* with laser → *HF* with laser → *DF* without laser → *HF* without laser.

### 6.2 Research limitations

This research has some limitations:

- The experiments were performed on only one person at a time and not in crowded environments (without the ability to distinguish between people).
- The obstacles and the entire room of the experiment was pre-tested and adapted to reduce false alarms of legs detection due to chair and table legs.
- Indoor environments with manually adapting light conditions depends on the sun reflection.

- The subjects and the robot moved slowly to test the robot's following parameters that depend on large computing complexity.

### 6.3 Future Work

The algorithms and following methods presented in this thesis were tested in real-time on a mobile robot for human-following in unknown environments. There are several suggestions for future research:

- In this research, the *CO* uses the *DO* (depth) algorithm when the person is close to the Kinect and the *VO* (vision) algorithm when the person is far from the Kinect. A better combined algorithm can be developed by using both of the sensors in parallel for comparing the results of each algorithm by defining a combined decision parameter for occlusion detection.
- When the occlusion detection algorithm detects an occlusion process, it changes the robot's following angle to  $15^\circ$  (small occlusion or wall occlusion) or  $30^\circ$  (large occlusions). Adjusting the angle according to the size of the occlusion can result in improved performance (instead of the current simple selection between two fixed angles).
- Future research should focus on the use of dynamic maps, using SLAM to help the robot navigate and orientate in previously visited environments.
- Implementing ability of distinguish between people by using people parameters like height and width, or clothes parameters like color and shape to distinguish between people and then to search for the particular person being followed after loss.
- Developing following angle above  $30^\circ$  (like side by side following).
- For obstacles detection, the vertical search field of view should be increased by using the Kinect; the currently used technique is based on legs detection by a laser sensor, which has a very narrow search field of view (20 cm above the ground).
- Improve the run time during *HF* by adding more parallel computers and increase the computing size.

## 7. References

- Alvarez, S., Pardo, M., Iglesias, R., Canedo, A., & Regueiro, V. (2012) Feature Analysis for Human Recognition and Discrimination : Application to a Person-Following Behaviour in a Mobile Robot. *Robotics and Autonomous Systems* 60 (8): 1021–1036.
- Young, E., Kamiyama, Y., Reichenbach, J., Igarashi, T., & Sharlin, E. (2011). How to walk a robot: A dog-leash human-robot interface. *IEEE Int. Work. Robot Human Interaction Communication*: 376–382.
- Ding, S., Gangdun, L., Yang, L., Zhang, J., & Yuan, J. (2015). SLAM and Moving Target Tracking Based on Constrained Local Submap Filter. *IEEE International Conference on Information and Automation*: 831–836.
- Doisy, G., Jevti, A., Lucet, E., & Edan, Y. (2012). Adaptive Person-Following Algorithm Based on Depth Images and Mapping. *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS Vilamoura, Portugal, October 7-12*.
- Garcia, A., Garcia, A., Hagaras, H., Dooley, J., Callaghan, V., & Botia, J. (2013). A Fuzzy Logic-Based System for Indoor Localization Using WiFi in Ambient Intelligent Environments. *IEEE Transactions on Fuzzy Systems* 21 (4): 702–718.
- Granata, C., Bidaud, P., & Buendia, A. (2011). Interactive Person Following for Social Robots: Hybrid Reasoning Based on Fuzzy and Multiple-Objectives Decision Making Approaches. *14th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*: 1–16.
- Hadi, H., Rosbi, M., Sheikh, U., & Amin, M. (2015). Improved Occlusion Handling for Human Detection from Mobile Robot. *Science and Information Conference (SAI)*: 694–698.
- Huang, Chang, & Haizhou. (2007). High-Performance Rotation Invariant Multiview Face Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29 (4): 671–686.
- Rlenmes, I., Karakaya, S., Küçüky, G., Ocak, H., & Bingül, Z. (2012). Obstacles and optimal heading direction detection algorithm on a mobile robot platform. *20th Signal Processing and Communications Applications Conference (SIU)*: 1-4.
- Ikemura, Sho, & Fujiyoshi. (2011). Real-Time Human Detection. *Computer Vision – ACCV 2010 Volume 6495*: 25–38.
- Jafari, O., Mitzel, D., & Leibe, B. (2014). Real-Time RGB-D Based People Detection and Tracking for Mobile Robots and Head-Worn Cameras. *IEEE International Conference on Robotics and Automation (ICRA)*: 5636–5643.
- Jia, S., Wang, S., Wang, L., & Li, X. (2016). Human Tracking System Based on Adaptive Multi- Feature Mean-Shift for Robot under the Double- Layer Locating Mechanism. *Advanced Robotics* 28 (24): 1653-1664.
- Jung, E., Lee, J., Yi, B., & Park, J. (2014). Development of a Laser-Range-Finder-Based Human Tracking and Control Algorithm for a Marathoner Service Robot. *IEEE/ASME Transactions on Mechatronics* 19 (6): 1963–1975.
- Karakaya, S., Küçüky, G., Toprak, C., & Ocak, H. (2014). Development of a Human Tracking Indoor Mobile Robot Platform. *16th International Conference on Mechatronics - Mechatronika (ME)*: 683-687.
- Kim, H., Chung, W., & Yoo, Y. (2010). Detection and Tracking of Human Legs for a Mobile Service Robot. *IEEE/ASME International Conference on Advanced Intelligent*

Mechatronics: 812–817.

- Kmiotek, Pawel, & Ruichek. (2008). Representing and Tracking of Dynamics Objects Using Oriented Bounding Box and Extended Kalman Filter. *IEEE Conference on Intelligent Transportation Systems (ITSC)*: 322–328.
- Li, X., Dick, A., Shen, C., Hengel, A., & Wang, H. (2013). Incremental Learning of 3D-DCT Compact Representations for Robust Visual Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35 (4): 863–881.
- Li, Y., Ding, S., Zhai, Q., Zheng, Y., & Xuan, D. (2015). Human Feet Tracking Guided by Locomotion Model. *IEEE International Conference on Robotics and Automation (ICRA)*: 2424–2429.
- Liu, X. (2004). Hand Gesture Recognition Using Depth Data. *Sixth IEEE International Conference on Automatic Face and Gesture Recognition*: 529–534.
- Ma, X., Hu, C., Dai, X., & Qian, K. (2008). Sensor Integration for Person Tracking and Following with Mobile Robot. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*: 3254–3259.
- Machida, E., Cao, M., Murao, T., Hashimoto, H., & Cao, M. (2012). Human Motion Tracking of Mobile Robot with Kinect 3D Sensor. *SICE Annual Conference (SICE)*: 2207–2211.
- Morales, Y., Satake, S., Huq, r., Glas, D., Kanda, T., & Hagita, N. (2012). How Do People Walk Side-By-Side ? – Using A Computational Model Of Human Behavior For A Social Robot. *7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*: 301–308.
- Motai, Y., Jha, s., & Kruse, D. (2012). Signal Processing : Image Communication Human Tracking from a Mobile Agent : Optical Flow and Kalman Filter Arbitration. *Signal Processing : Image Communication* 27 (1): 83–95.
- Munaro, M., Horn, A., Illum, R., Burke, J., & Bogdan, R. (2014). OpenPTrack: People Tracking for Heterogeneous Networks of Color-Depth Cameras. *1st Intl. Workshop on 3D Robot Perception with Point Cloud Library at the 13th Intelligent Autonomous Systems Conference (IAS-13)*: 1–13.
- Munaro, M., & Menegatti, E. (2014). Fast RGB-D People Tracking for Service Robots. *Autonomous Robots* 37: 227–242.
- Najmaei, N., & Kermani, M. (2011). Applications of Artificial Intelligence in Safe Human – Robot Interactions. *IEEE Trans Syst Man Cybern B Cybern* 41 (2): 448–459.
- Norhidayah & Norida. (2015). Particle Filter in simultaneous localization and mapping (SLAM) using differential drive mobile. *Journal Teknologi*: 20: 91–97.
- Ota, M., Ogitsu, t., Hisahara, H., Takemura, H., Ishii, Y., & Mizoguchi, H. (2013). Recovery Function for Human Following Robot Losing Target. *IECON 39th Industrial Electronics Conference*: 4253–4257.
- Plagemann, C. (2010). Real Time Motion Capture Using a Single Time-Of-Flight Camera. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*: 755–762.
- Plagemann, C., & Koller, D. (2010). Real-Time Identification and Localization of Body Parts from Depth Images. *IEEE International Conference on Robotics and Automation (ICRA)*: 3108–3113.
- Pucci, D., Marchetti, L., & Morin, P. (2013). Nonlinear Control of Unicycle-like Robots for

- Person Following. IEEE/RSJ International Conference on Intelligent Robots and Systems, 3406–3411.
- Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, j., Berger, E., Wheeler, r., & Mg, A. (2009). ROS: An Open-Source Robot Operating System. ICRA workshop on open source software 3 (1): 5.
- Rusu, R., & Cousins, S. (2011). 3D Is Here: Point Cloud Library (PCL). IEEE International Conference on Robotics and Automation (ICRA): 1–4.
- Sahoo, S., & Ari, S. (2015). Automated Human Tracking Using Advanced Mean Shift Algorithm. International Conference on Communications and Signal Processing (ICCSP): 789–793.
- Sales, J., Cervera, E., & Baynat, V. (2010). Multi-Sensor Person Following in Low-Visibility Scenarios. Sensors 10 (12): 10953–10966.
- Schmidt, A. (2016). Prediction-Based Perspective Warping SLAM Accuracy. Springer International Publishing In Man–Machine Interactions 4: 169–177.
- Yao, A., Lin, X., Wang, G., & Yu, S. (2012). A Compact Association of Particle Filtering and Kernel Based Object Tracking. Pattern Recognition 45 (7): 2584–2597.

## Appendix A- An explanation of how to start the following methods and integrated algorithms

Connect to LXRobot Wifi

Computer 1 on the robot and Computer 2 not plugged to the robot

Computer 2- Open new terminal:

- `ssh robot@robot-desktop` (password: robot)
- `cd ~/workspace/ros/catkin`
- `roslaunch start_base.launch`

Computer 1- connect the kinect and the pan to computer 1 (on the robot)

Computer 1- open a terminal:

- `export ROS_MASTER_URI=http://robot-desktop:11311`
- `cd ~/workspace/ros/catkin/devel/lib/kinect2_bridge`
- `sudo ./kinect2_bridge` (ignore error)
- `roslaunch tracking_detection_and_tracking_kinect2.launch`

Computer 1- open a terminal:

- `export ROS_MASTER_URI=http://robot-desktop:11311`
- `roslaunch rosserial_python serial_node.py _port:=/dev/ttyUSB0 _baud:=250000`

Computer 1- open a terminal:

- `export ROS_MASTER_URI=http://robot-desktop:11311`
- `rostopic pub /Start_Stop_Pan std_msgs/Bool true`

Computer 1- open a terminal:

- `export ROS_MASTER_URI=http://robot-desktop:11311`
- `roslaunch kinect_orientation_control kinect_orientation_control_node`

The main following method can works with "leg\_detector", with Kinect detection ("Open\_PTrack") and even with both of them (priority to the Kinect and than the Laser Leg-Detector).

Computer 2- For "legs detector"- in new terminal:

- `cd workspace/ros/catkin/src/launchers`
- `roslaunch start_leg_detector.launch`



**Computer 2-** For obstacles avoidance, search for obstacles with the laser in real time- in new terminal:

- **roslaunch obstacles laser\_obstacles\_avoidance**

**Computer 1-** For occlusion detection, declares an occlusion using the depth information of the Kinect during person detection- in new terminal:

- **roslaunch occlusions depth\_occlusions**

There are 4 main methods codes for robot following with integrated unknown environments algorithms. Each one of them can work by himself.

In new terminal run one of the following lines:

**computer 2-** in new terminal:

- direct following with search algorithm when the person disappears:
  - **roslaunch people\_follower simple\_follower\_kinect2\_pan\_laser**
- direct following without search algorithm when the person disappears:
  - **roslaunch people\_follower  
simple\_follower\_kinect2\_pan\_laser\_without\_search**
- history following with search algorithm when the person disappears:
  - **roslaunch people\_follower path\_follower**
- history following without search algorithm when the person disappears:
  - **roslaunch people\_follower path\_follower\_without\_search**

## Appendix B- Likert-Style Questions for Section 3.4.3

After each trial (back-following and side-following):

Stressed by the task	1	2	3	4	5
Stressed by the robot	1	2	3	4	5
Person adapted behavior based on robot	1	2	3	4	5
Robot adapted behavior based on person	1	2	3	4	5
Walking was independent of robot	1	2	3	4	5
Walking was comfortable with speed of robot	1	2	3	4	5
Robot moved too slowly	1	2	3	4	5
Person was satisfied with the quality of following	1	2	3	4	5
Person felt safe regarding the distance of the robot	1	2	3	4	5
I lowered my speed to adapt to the speed of the robot	1	2	3	4	5

Final questionnaire to compare the two trials and the opinion on the robot:

I felt a difference between the two trials	1	2	3	4	5
In which trial did you feel more comfortable?	1		2		
Why					
<b>Opinion of the Robot</b> (the robot is..)					
Friendly	1	2	3	4	5
Disturbing	1	2	3	4	5
Considerate	1	2	3	4	5
Dangerous	1	2	3	4	5
Scary	1	2	3	4	5
Annoying	1	2	3	4	5
Stressful	1	2	3	4	5

## Appendix C- Statistical Analysis

For the analysis of the results for the comparison between occlusion detection algorithms (section 5.4), the Direct-Following experiment (section 5.5), the History-Following experiment (section 5.6) and the Adaptive vs Non-Adaptive Direct-Following and History-Following experiment (section 5.7) there is a necessity to compare the means of the groups. The comparisons are performed using one-way ANOVA (analysis of variance) and Tukey's HSD test. The ANOVA analysis employs an F-test to determine whether there is a significant difference between two or more of the means. Tukey's HSD is a post hoc multiple comparison test, performed after the F-test determines that the means are not equal. The Tukey's HSD test separates and ranks the groups demanding 95% confidence level ( $\alpha=0.05$ ) for the entire comparison.

### D.1 Occlusion's algorithms comparison

Raw data:

*Table 20 - Raw data occlusion's algorithms comparison*

type	distance	true_occlusion	true_wall	true_small	true_big	false_occlusion	false_wall	false_small	false_big
depth	2	0.97	1	0.12	0.85	0	0	0	0
rgb	1.98		0				0		
depthrgb	2.07	0.49	0.5	0.04	0.44	0	0	0	0
depth	3.63	0.94	0.92	0.14	0.79	0.44	0	0.44	0
rgb	4.05		0				0.67		
depthrgb	4.18	0.45	0.44	0.05	0.41	0.29	0	0.29	0
depth	4.9	0.72	0.73	0.03	0.68	0.28	0	0.05	0.23
rgb	4.89		0.11				0.1		
depthrgb	4.71	0.35	0.35	0.01	0.34	0.13	0	0.02	0.11
depth	5.81	0.92	0.91	0.12	0.8	0	0	0	0
rgb	5.79		0				0.79		
depthrgb	5.7		0.02				0.18		
depth	7.95	1	1	0.03	0.97	0	0	0	0
rgb	7.94		0.42				0.27		
depthrgb	7.83		0.13				0.18		
depth	4.33	0.95	0	0.7	0.25	0	0	0	0
rgb	3.94		0.17				0		
depthrgb	4.29	0.5	0	0.24	0.26	0	0	0	0

Statistical analysis (ANOVA and Tukey):

*Table 21- ANOVA occlusion's algorithms comparison*

ANOVA						
		Sum of Squares	df	Mean Square	F	Sig.
total_true	Between Groups	1.722	2	.861	25.748	.000
	Within Groups	.502	15	.033		
	Total	2.224	17			
trueWall	Between Groups	1.415	2	.708	9.764	.002
	Within Groups	1.087	15	.072		
	Total	2.502	17			
total_false	Between Groups	.147	2	.073	1.980	.173
	Within Groups	.556	15	.037		
	Total	.703	17			

*Table 22- Tukey HSD occlusion's algorithms comparison*

Multiple Comparisons							
Tukey HSD							
Dependent Variable	(I) algorithm	(J) algorithm	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
						Lower Bound	Upper Bound
total_true	do	vo	.72167*	.10559	.000	.4474	.9959
		co	.56083*	.10559	.000	.2866	.8351
	vo	do	-.72167*	.10559	.000	-.9959	-.4474
		co	-.16083	.10559	.308	-.4351	.1134
	co	do	-.56083*	.10559	.000	-.8351	-.2866
		vo	.16083	.10559	.308	-.1134	.4351
trueWall	do	vo	.64333*	.15543	.002	.2396	1.0471
		co	.53000*	.15543	.010	.1263	.9337
	vo	do	-.64333*	.15543	.002	-1.0471	-.2396
		co	-.11333	.15543	.750	-.5171	.2904
	co	do	-.53000*	.15543	.010	-.9337	-.1263
		vo	.11333	.15543	.750	-.2904	.5171
total_false	do	vo	-.20667	.11118	.185	-.4955	.0821
		co	-.03500	.11118	.947	-.3238	.2538
	vo	do	.20667	.11118	.185	-.0821	.4955
		co	.17167	.11118	.299	-.1171	.4605
	co	do	.03500	.11118	.947	-.2538	.3238
		vo	-.17167	.11118	.299	-.4605	.1171

\*. The mean difference is significant at the 0.05 level.

Table 23- Total\_True occlusion's algorithms comparison

total_true			
Tukey HSD <sup>a</sup>			
algorithm	N	Subset for alpha = 0.05	
		1	2
vo	6	.1167	
co	6	.2775	
do	6		.8383
Sig.		.308	1.000

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 6.000.

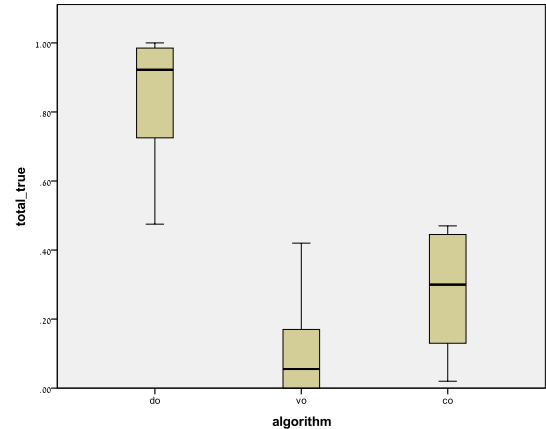


Table 24- True\_Wall occlusion's algorithms comparison

trueWall			
Tukey HSD <sup>a</sup>			
algorithm	N	Subset for alpha = 0.05	
		1	2
vo	6	.1167	
co	6	.2300	
do	6		.7600
Sig.		.750	1.000

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 6.000.

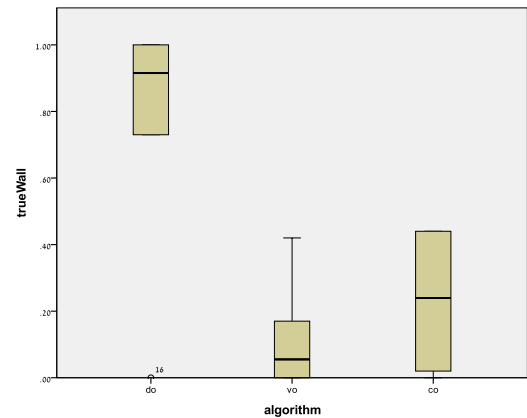
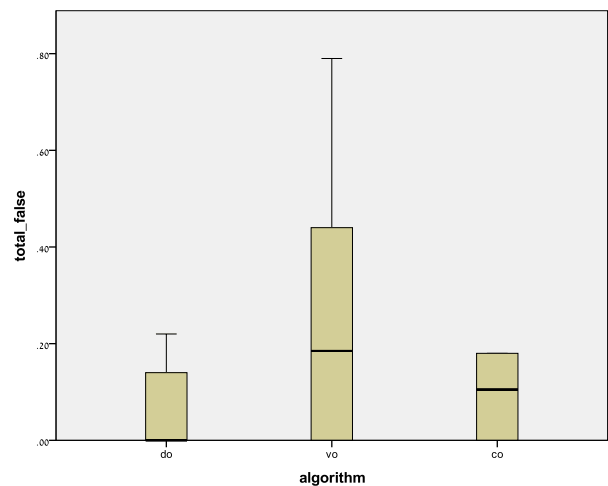


Table 25- Total\_False occlusion's algorithms comparison

total_false		
Tukey HSD <sup>a</sup>		
algorithm	N	Subset for alpha = 0.05
		1
do	6	.0600
co	6	.0950
vo	6	.2667
Sig.		.185

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 6.000.



Correlations (Pearson):

The Pearson product-moment correlation coefficient is a measure of the linear correlation between two variables X and Y, giving a value between +1 and -1 inclusive, where 1 is total positive correlation, 0 is no correlation and -1 is total negative correlation.

*Table 26- PEARSON correlations occlusion's algorithms comparison*

Correlations				Correlations			
		do_result	co_result			co_result	vo_result
do_result	Pearson Correlation	1	.706**	co_result	Pearson Correlation	1	.535**
	Sig. (2-tailed)		.000		Sig. (2-tailed)		.009
	N	45	45		N	44	23
co_result	Pearson Correlation	.706**	1	vo_result	Pearson Correlation	.535**	1
	Sig. (2-tailed)	.000			Sig. (2-tailed)	.009	
	N	45	45		N	23	23

\*\* . Correlation is significant at the 0.01 level (2-tailed).

\*\* . Correlation is significant at the 0.01 level (2-tailed).

## D.2 Direct Following and History Following Experiments

Raw data:

Table 27- Raw data Direct-Following and History-Following

	subject	trial	direct0 /history1	order	total loss	loss with self recover	loss with intervention	safety intervention	obstacles hit	avarage distance	std distance	ratio track/no track	false alarm depth occlusion	depth occlusion	avarage velocity of the person	robot distance
1	1	1	0	1	0	0	0	0	1	3.2	0.78	0.99	-	-	0.55	14.34
2	1	2	0	2	1	1	0	0	0	2.96	0.74	0.95	0.3	0.8	0.61	10.51
3	1	3	0	3	1	0	1	0	0	2.98	0.47	0.72	0.6	0.7	0.53	16.94
4	1	4	1	4	1	0	1	0	0	3.52	0.85	0.72	-	-	0.45	18.19
5	1	5	1	5	1	0	1	0	0	3.63	0.75	0.82	0.6	0.7	0.48	19.54
6	2	1	0	3	0	0	0	0	1	3.23	0.92	1	-	-	0.6	11.78
7	2	2	0	5	1	1	0	0	1	2.84	0.69	0.98	0.6	0.7	0.67	10.79
8	2	3	0	4	2	0	2	1	0	3.14	0.91	0.73	0.6	0.8	0.58	11.95
9	2	4	1	2	1	0	1	1	0	3.07	0.77	0.8	-	-	0.66	15.7
10	2	5	1	1	2	1	1	1	0	3.76	0.86	0.61	0.6	0.7	0.58	20.41
11	3	1	0	4	1	1	0	0	0	3.04	0.88	0.98	-	-	0.57	13.53
12	3	2	0	3	0	0	0	0	0	3.23	0.84	1	0.5	0.9	0.56	12.39
13	3	3	0	5	0	0	0	0	1	3.47	0.79	0.99	0.5	0.8	0.59	14
14	3	4	1	1	3	0	3	0	0	3.52	0.73	0.54	-	-	0.51	16.65
15	3	5	1	2	1	0	1	1	0	3.67	0.82	0.8	0.6	0.8	0.58	18.56
16	4	1	0	2	0	0	0	1	0	2.71	0.64	1	-	-	0.41	16.39
17	4	2	0	3	1	1	0	0	0	2.7	0.82	1	0.5	0.9	0.42	12.87
18	4	3	0	1	0	0	0	0	0	2.81	0.69	1	0.4	0.8	0.42	14
19	4	4	1	5	0	0	0	0	0	3.14	1.33	1	-	-	-	12.99
20	4	5	1	4	0	0	0	0	0	3.38	0.93	1	0.3	0.9	0.39	19.11
21	5	1	0	3	0	0	0	0	0	2.45	0.72	0.99	-	-	0.45	12.63
22	5	2	0	1	0	0	0	0	0	2.74	0.54	0.99	0.3	0.8	0.48	13.63
23	5	3	0	2	0	0	0	0	0	2.96	0.73	0.98	0.3	0.8	0.42	13.94
24	5	4	1	4	0	0	0	0	1	3.17	1	1	-	-	0.39	15.51
25	5	5	1	5	1	0	1	1	0	2.83	1.18	0.91	0.4	0.9	0.38	17.23
26	6	1	0	5	1	1	0	2	0	4.01	1.21	0.96	-	-	0.56	12.22
27	6	2	0	4	0	0	0	1	0	3.17	0.6	1	0.5	0.9	0.52	17.02
28	6	3	0	3	3	0	3	1	0	2.8	0.65	0.7	0.5	0.8	0.54	10.26
29	6	4	1	2	2	1	1	0	0	3.27	0.96	0.84	-	-	0.47	17.84
30	6	5	1	1	2	1	1	0	0	3.74	1.03	0.81	0.5	0.8	0.43	20.29
31	7	1	0	3	0	0	0	0	0	3.33	0.67	0.97	-	-	0.66	12.05
32	7	2	0	4	1	1	0	0	1	3.27	0.73	0.96	0.5	0.9	0.65	11.99
33	7	3	0	5	2	0	2	1	0	4.9	1.8	0.55	0.5	0.8	0.59	15.86
34	7	4	1	1	1	0	1	1	0	3.33	0.77	0.77	-	-	0.5	15.72
35	7	5	1	2	2	1	1	1	1	3.47	0.83	0.8	0.6	0.8	0.52	18.96

Statistical analysis **Direct** Following Experiment (ANOVA and Tukey):

Table 28- ANOVA Ratio\_Track\_Kinect Direct-Following

### ANOVA

Ratio_Track_Kinect					
	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	1406.000	2	703.000	3.633	.047
Within Groups	3483.143	18	193.508		
Total	4889.143	20			

Table 29- Tukey HSD Ratio\_Track\_Kinect Direct-Following

Ratio\_Track\_Kinect  
Tukey HSD

(I) trial	(J) trial	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
1	2	.14286	7.43559	1.000	-18.8340	19.1197
	3	17.42857	7.43559	.075	-1.5483	36.4054
2	1	-.14286	7.43559	1.000	-19.1197	18.8340
	3	17.28571	7.43559	.078	-1.6911	36.2626
3	1	-17.42857	7.43559	.075	-36.4054	1.5483
	2	-17.28571	7.43559	.078	-36.2626	1.6911

Table 30- Ratio\_Track\_Kinect Direct-Following

Ratio\_Track\_Kinect

Tukey HSD<sup>a</sup>

trial	N	Subset for alpha = 0.05
		1
3	7	81.0000
2	7	98.2857
1	7	98.4286
Sig.		.075

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 7.000.

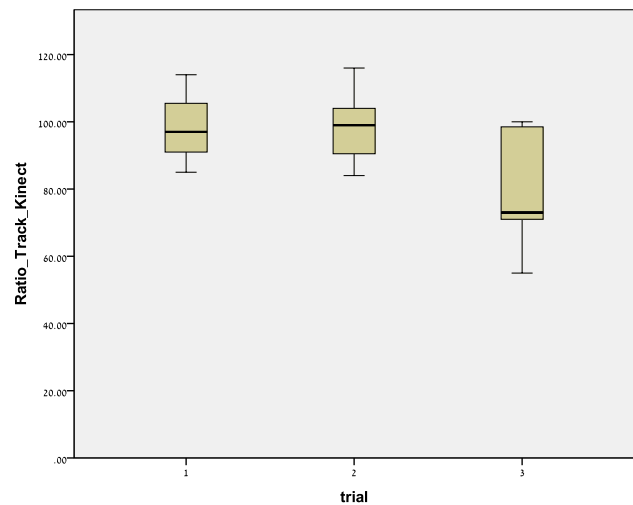


Table 31- Descriptive Total\_Loss Direct-Following

Descriptives

total loss

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
no search	7	1.14	1.215	.459	.02	2.27	0	3
search	14	.43	.514	.137	.13	.73	0	1
Total	21	.67	.856	.187	.28	1.06	0	3



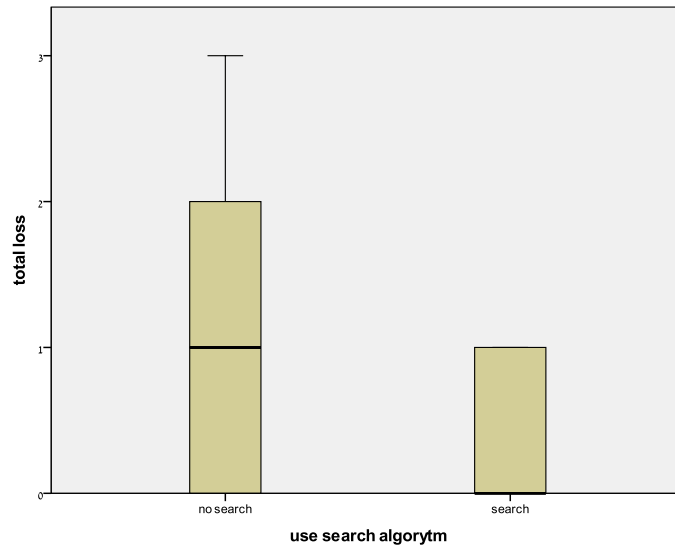


Table 32- ANOVA Subject's\_average\_velocity Direct-Following

**ANOVA**

average velocity of the subject

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	.119	6	.020	19.860	.000
Within Groups	.014	14	.001		
Total	.133	20			

Table 33- Tukey HSD Subject's\_average\_velocity Direct-Following

**average velocity of the subject**

Tukey HSD<sup>a</sup>

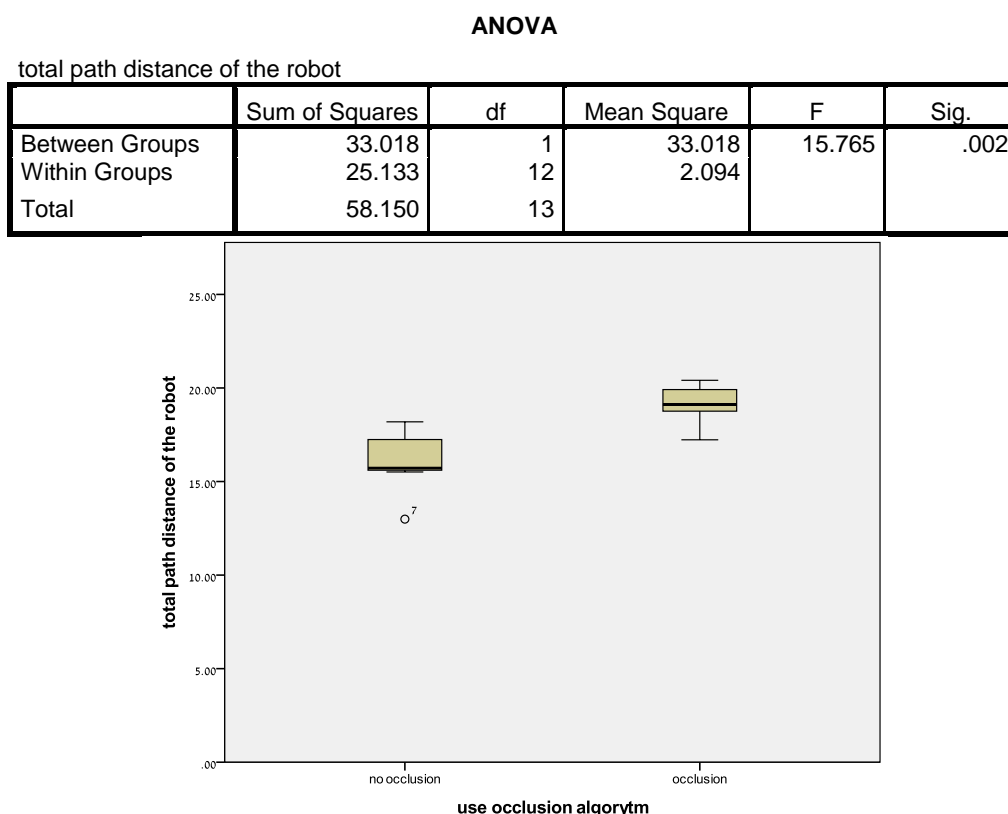
subject	N	Subset for alpha = 0.05		
		1	2	3
4	3	.4167		
5	3	.4500		
6	3		.5400	
1	3		.5633	.5633
3	3		.5733	.5733
2	3		.6167	.6167
7	3			.6333
Sig.		.844	.107	.164

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 3.000.

Statistical analysis **History** Following Experiment (ANOVA and Tukey):

*Table 34- ANOVA Total\_path\_distance History-Following*



Statistical analysis **Direct vs History** Following Experiment (ANOVA and Tukey):

Levene's test:

Levene's test is an inferential statistic used to assess the equality of variances for a variable calculated for two or more groups. It tests the null hypothesis that the population variances are equal. If the resulting p-value is less than significance level of 0.05, the obtained differences in sample variances are unlikely to have occurred based on random sampling from a population with equal variances.

*Table 35- Homogeneity of Variances Direct vs History*

Test of Homogeneity of Variances				
	Levene Statistic	df1	df2	Sig.
total loss	.001	1	33	.973
percent of intervent recover from total loss	.395	1	33	.534
ratio track o track	.001	1	33	.973
total path distance of the robot	.076	1	33	.785
average velocity of the subject	.000	1	32	.989

*Table 36- ANOVA Direct vs History*

		ANOVA				
		Sum of Squares	df	Mean Square	F	Sig.
total loss	Between Groups	2.519	1	2.519	3.322	.077
	Within Groups	25.024	33	.758		
	Total	27.543	34			
percent of intervent recover from total loss	Between Groups	1.719	1	1.719	10.404	.003
	Within Groups	5.452	33	.165		
	Total	7.171	34			
ratio track o track	Between Groups	.102	1	.102	5.804	.022
	Within Groups	.578	33	.018		
	Total	.679	34			
total path distance of the robot	Between Groups	157.595	1	157.595	37.868	.000
	Within Groups	137.337	33	4.162		
	Total	294.931	34			
avarage velocity of the subject	Between Groups	.024	1	.024	3.484	.071
	Within Groups	.217	32	.007		
	Total	.240	33			

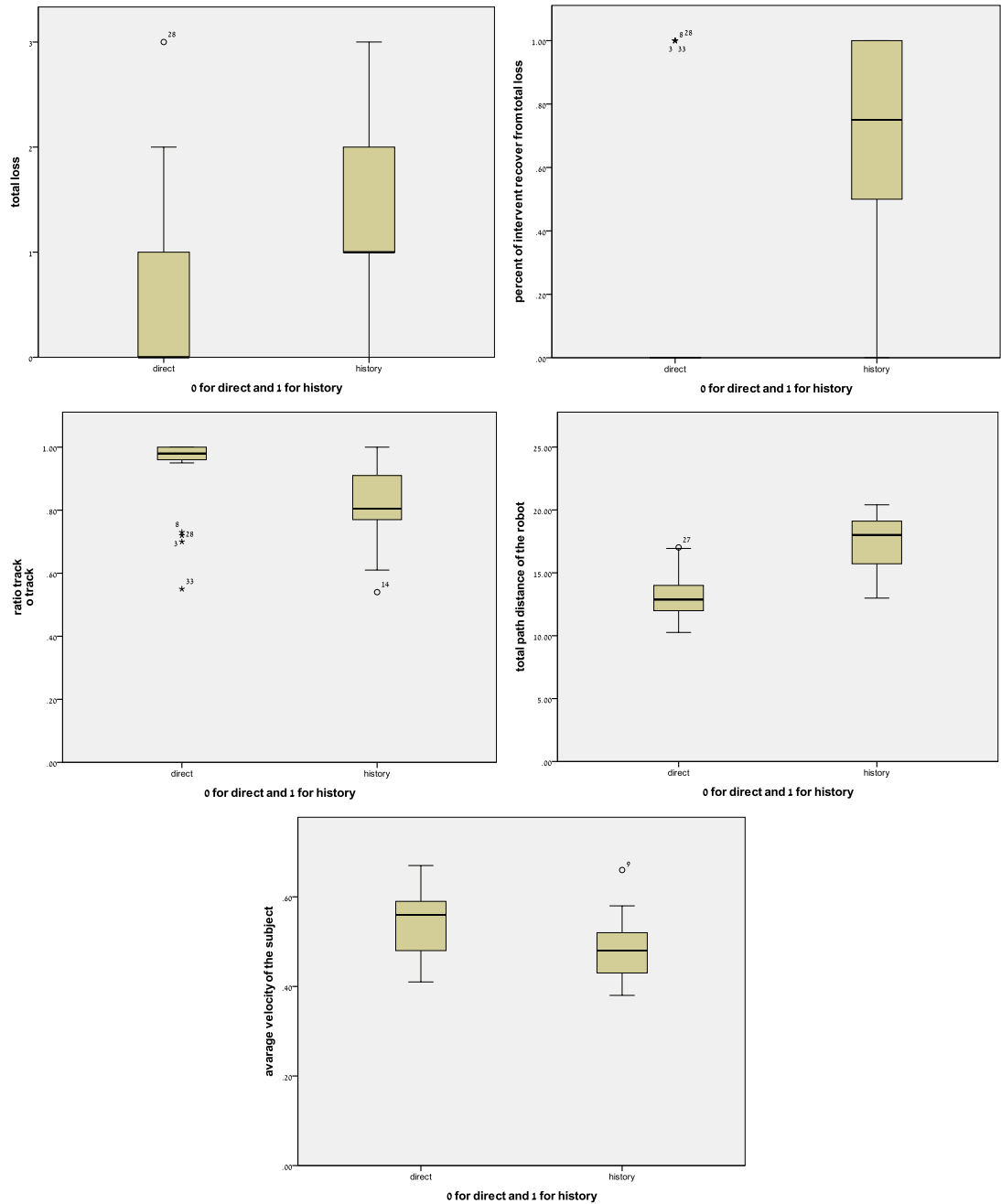


Figure 29- Descriptive Direct vs History

Statistical analysis **5 trials of Direct (2) and History (3)** Following Experiment (ANOVA and Tukey):

Table 37- Homogeneity of Variances Total\_path\_distance- 5 trials

Test of Homogeneity of Variances				
total path distance of the robot				
Levene Statistic	df1	df2	Sig.	
.494	4	30	.740	

Table 38- ANOVA Total\_path\_distance- 5 trials

ANOVA					
total path distance of the robot					
	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	194.904	4	48.726	14.614	.000
Within Groups	100.027	30	3.334		
Total	294.931	34			

Table 39- Tukey HSD Total\_path\_distance- 5 trials

total path distance of the robot  
Tukey HSD

(I) trial	(J) trial	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
1	2	.53429	.97603	.981	-2.2968	3.3654
	3	-.57286	.97603	.976	-3.4039	2.2582
	4	-2.80857	.97603	.053	-5.6397	.0225
	5	-5.88000*	.97603	.000	-8.7111	-3.0489
2	1	-.53429	.97603	.981	-3.3654	2.2968
	3	-1.10714	.97603	.787	-3.9382	1.7239
	4	-3.34286*	.97603	.014	-6.1739	-.5118
	5	-6.41429*	.97603	.000	-9.2454	-3.5832
3	1	.57286	.97603	.976	-2.2582	3.4039
	2	1.10714	.97603	.787	-1.7239	3.9382
	4	-2.23571	.97603	.176	-5.0668	.5954
	5	-5.30714*	.97603	.000	-8.1382	-2.4761
4	1	2.80857	.97603	.053	-.0225	5.6397
	2	3.34286*	.97603	.014	.5118	6.1739
	3	2.23571	.97603	.176	-.5954	5.0668
	5	-3.07143*	.97603	.028	-5.9025	-.2403
5	1	5.88000*	.97603	.000	3.0489	8.7111
	2	6.41429*	.97603	.000	3.5832	9.2454
	3	5.30714*	.97603	.000	2.4761	8.1382
	4	3.07143*	.97603	.028	.2403	5.9025

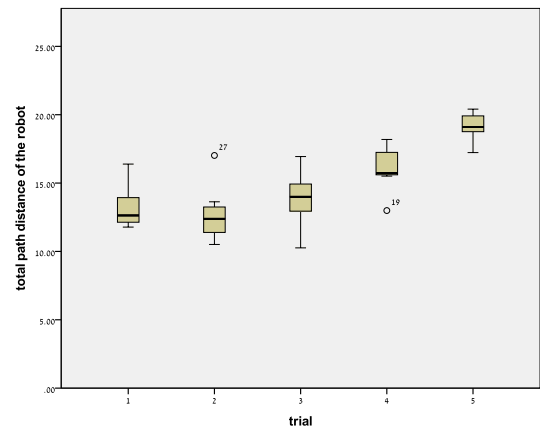
\*. The mean difference is significant at the 0.05 level.

Table 40- Total\_path\_distance- 5 trials

total path distance of the robot				
Tukey HSD <sup>a</sup>				
trial	N	Subset for alpha = 0.05		
		1	2	3
2	7	12.7429		
1	7	13.2771	13.2771	
3	7	13.8500	13.8500	
4	7		16.0857	
5	7			19.1571
Sig.		.787	.053	1.000

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 7.000.



## D.3 Adaptive Kinect-Laser Direct and History Following Experiment

Raw data:

Table 41- Raw data Adaptive vs Non-Adaptive (Direct and History)

	subject	gender	male/female	direct0/history1	without0/withLaser1	order	total loss	loss with self recover	loss with intervention	safety intervention	obstacles hit	height	laser obstacles	matce	avarage distance	std distance	legs false alarm (percent from ratio)	ratio track/no track laser	ratio track/no track kinect	false alarm depth occlusion	depth occlusion	avarage velocity of the person	robot distance	kinect fall
1	1	0	1	1	1	1	2	1	1	3	1	1.7	62	7	3.97	0.84	0.95	0.132	0.983	0.25	0.8	0.45	29.246	
2	1	0	2	1	0	2	1	0	1	0	0	1.66	11		4.35	0.94			0.812	0.15	0.8	0.5	31.383	1
3	1	0	3	0	1	3	1	1	0	2	0	1.69	67	4	3.61	0.75	0.4	0.52	0.94	0.1	0.8	0.6	18.614	
4	1	0	4	0	0	4	2	2	0	2	0	1.67	48		3.67	0.81			0.95	0.25	0.75	0.6	20.681	
5	2	1	1	1	1	1	2	2	0	1	0	1.53	57	0	3.59	0.81	1	0.03	0.95	0.25	0.75	0.3	28.677	
6	2	1	2	1	0	2	1	0	1	1	0	1.55	43		4.02	1.06			0.94	0.3	0.7	0.3	31.831	
7	2	1	3	0	1	4	1	1	0	1	0	1.61	51	0	4.45	1.47	1	0.57	0.98	0.35	0.65	0.4	25.073	
8	2	1	4	0	0	3	2	0	2	1	1	1.54	38		3.5	1.07			0.95	0.25	0.8	0.4	25.521	
9	3	0	1	1	1	1	2	2	0	0	1	1.7	56	0	3.69	1.05	1	0.04	0.96	0.4	0.8	0.3	20.767	
10	3	0	2	1	0	3	1	0	1	2	0	1.71	35		3.4	1.05			0.79	0.4	0.9	0.3	28.623	1
11	3	0	3	0	1	2	1	1	0	1	0	1.69	30	0	3.3	0.56		0	0.96	0.45	0.9	0.5	18.967	
12	3	0	4	0	0	4	1	0	1	1	0	1.65	0		4.05	1.07			0.92	0.4	0.8	0.5	18.165	
13	4	1	1	1	1	1	2	0	1	2	0	1.47	62	4	3.76	0.93	0.95	0.14	0.95	0.3	0.75	0.4	23.545	
14	4	1	2	1	0	3	1	2	3	0	0	1.56	92		4.1	0.95			0.82	0.5	0.8	0.4	27.568	
15	4	1	3	0	1	4	2	2	0	1	0	1.5	19	0	3.31	0.79	1	0.1	0.92	0.2	0.9	0.5	18.416	
16	4	1	4	0	0	2	2	0	2	2	0	1.5	20		3.6	1.26			0.94	0.2	0.9	0.5	21.213	
17	5	1	1	1	1	1	2	1	1	2	1	1.62	35	22	3.68	0.95	0.4	0.3	0.93	0.2	0.9	0.2	20.277	
18	5	1	2	1	0	4	3	0	3	3	0	1.61	36		4.34	1.09			0.9	0.5	0.8	0.4	19.866	
19	5	1	3	0	1	2	2	0	2	1	0	1.64	41	2	3.83	0.54	0.4	0.46	0.75	0.05	0.8	0.5	16.775	1
20	5	1	4	0	0	3	0	0	0	0	0	1.63	39		3.46	0.74			1	0.25	0.7	0.5	17.881	
21	6	0	1	1	1	1	2	1	1	0	0	1.62	59	9	3.76	0.97	0.4	0.37	0.87	0.3	0.8	0.3	22.046	
22	6	0	2	1	0	4	3	1	2	2	1	1.61	71		3.68	0.69			0.94	0.4	0.8	0.3	22.582	
23	6	0	3	0	1	3	1	1	0	0	0	1.61	89	15	3.56	0.8	0.1	0.24	0.97	0.2	0.9	0.4	21.833	1
24	6	0	4	0	0	2	0	0	0	0	0	1.62	38		3.19	1.05			0.98	0.5	0.5	0.4	21.977	
25	7	0	1	1	1	2	2	2	0	0	0	1.71	71	49	3.38	0.73	0.5	0.53	0.86	0.3	0.8	0.2	26.562	
26	7	0	2	1	0	1	3	1	2	2	0	1.69	71		4.06	1.35			0.91	0.2	0.7	0.2	27.789	
27	7	0	3	0	1	3	1	1	0	2	0	1.71	52	37	2.88	0.97	0	0.82	0.97	0.1	0.8	0.3	17.009	
28	7	0	4	0	0	4	2	0	2	3	0	1.74	14		2.05	0.33			0.85	0.1	0.1	0.3	24.368	
29	8	0	1	1	1	2	3	1	2	0	0	1.65	76	7	4.86	1.31	0.2	0.26	0.45	0.4	0.1	0.4	26.67	1
30	8	0	2	1	0	1	2	1	1	1	0	1.67	73		3.94	1.15			0.92	0.2	0.6	0.4	22.404	
31	8	0	3	0	1	4	0	0	0	1	0	1.68	50	34	2.95	0.51	0.1	0.71	0.99	0.15	0.8	0.4	17.448	
32	8	0	4	0	0	3	2	0	2	1	1	1.68	57		2.82	0.44			0.83	0.35	0.8	0.4	18.308	
33	9	1	1	1	1	2	1	1	0	1	0	1.7	0	15	3.67	0.88	0.4	0.75	0.93	0.25	0.8	0.3	23.591	
34	9	1	2	1	0	3	4	1	3	2	1	1.7	54		3.71	0.94			0.83	0.2	0.8	0.3	20.255	
35	9	1	3	0	1	1	0	0	0	0	0	1.66	30	41	3.01	0.66	0.1	0.43	0.98	0.1	0.9	0.3	18.718	
36	9	1	4	0	0	4	3	0	3	2	0	1.62	16		4.41	0.78			0.59	0.7	0.6	0.3	21.452	1
37	10	0	1	1	1	2	3	1	2	1	0	1.57	20	7	3.52	0.64	0.5	0.66	0.85	0.2	0.7	0.3	25.893	
38	10	0	2	1	0	3	4	2	2	1	0	1.91	19		3.91	1.05			0.94	0.4	0.8	0.3	27.867	
39	10	0	3	0	1	4	0	0	0	0	0	1.62	10	60	2.78	0.47	0.1	0.86	1	0.1	0.9	0.4	19.315	
40	10	0	4	0	0	1	4	1	3	1	0	1.62	49		3.68	1.11			0.84	0.3	0.8	0.4	18.252	
41	11	0	1	1	1	2	2	1	1	1	0	1.63	58	3	3.7	1.28	0.6	0.48	0.73	0.2	0.5	0.4	27.175	
42	11	0	2	1	0	4	3	1	2	1	0	1.59	47		4.05	0.83			0.96	0.3	0.7	0.4	25.282	
43	11	0	3	0	1	1	0	0	0	1	0	1.6	71	56	2.65	0.54	0.6	0.75	0.99	0.2	0.8	0.4	19.902	
44	11	0	4	0	0	3	0	0	0	2	0	1.6	85		3.21	0.89			0.99	0.2	0.7	0.4	22.059	
45	12	0	1	1	1	2	1	1	0	2	0	1.64	78	35	3.43	0.88	0	0.27	0.96	0.2	0.8	0.3	20.045	
46	12	0	2	1	0	4	4	1	3	2	1	1.63	84		3.37	1.05			0.77	0.7	0.7	0.3	20.097	
47	12	0	3	0	1	3	0	0	0	0	0	1.66	23	74	2.42	0.38	0	0.96	0.99	0.15	0.7	0.3	20.969	
48	12	0	4	0	0	1	3	1	2	1	0	1.66	23		2.36	0.63			0.92	0.2	0.6	0.3	20.622	
49	13	1	1	1	1	3	0	0	0	1	0	1.55	59	40	3.31	0.97	0.05	0.29	0.95	0.2	0.7	0.3	21.345	
50	13	1	2	1	0	1	4	0	4	1	0	1.56	74		3.77	0.78			0.87	0.4	0.8	0.3	21.261	
51	13	1	3	0	1	2	0	0	0	0	0	1.55	8	106	2.39	0.28	0	0.89	1	0.1	0.9	0.3	19.841	
52	13	1	4	0	0	4	3	1	2	2	0	1.57	45		3.17	1.07			0.96	0.3	0.7	0.3	19.334	
53	14	0	1	1	1	3	1	0	1	0	0	1.67	33	2	3.57	1.01	0.5	0.28	0.81	0.3	0.8	0.4	21.117	1
54	14	0	2	1	0	1	2	0	2	1	0	1.6	21		3.71	0.66			0.82	0.6	0.8	0.4	21.992	
55	14	0	3	0	1	4	1	1	0	1	0	1.65	65	38	3.11	1.04	0	0.56	0.87	0.2	0.9	0.4	17.382	
56	14	0	4	0	0	2	0	0	0	0	1	1.62	26		2.43	0.3			1	0.2	0.6	0.4	19.219	
57	15	0	1	1	1	3	2	0	2	1	0	1.61	53	7	3.9	0.9	0.5	0.22	0.71	0.3	0.8	0.4	20.954	
58	15	0	2	1	0	2	3	1	2	2	1	1.57	58		3.98	1.07			0.97	0.3	0.8	0.4	21.751	
59	15	0	3	0	1	1	1	1	0	1	0	1.6	27	42	2.92	0.83	0	0.65	1	0.3	0.8	0.4	18.464	
60	15	0	4	0	0	4	0	0	0	0	0	1.61	45		2.99	0.52			1	0.2	0.7	0.4	20.397	
61	16	0	1	1	1	3	1	1	0	0	0	1.52	46	12	3.23	0.64	0.3	0.18	0.98	0.2	0.7	0.3	23.084	
62	16	0	2	1	0	2	2	1	1	1	1	1.5	40		3.66	0.81			0.93	0.1	0.7	0.3	23.706	
63	16	0	3	0	1	4	1	1	0	1	0	1.52	27	4	2.88	0.55	0.3	0.47	0.95	0.3	0.9	0.4	16.641	
64	16	0	4	0	0	1	1	1	0	1	1	1.53	35		2.91	0.49			0.98	0.3	0.8	0.5	17.589	
65	17	1	1	1	1	3	2	1	1	1	0	1.68	37	23	3.6	0.75	0.2	0.28	0.9	0.2	0.8	0.4	20.751	
66	17	1	2	1	0	4	3	2	1	1	1	1.69	24		3.67	1.13			0.93	0.2	0.8	0.4	22.149	
67	17	1	3	0	1	1	1	1	0	1	0	1.7	30	8	2.71	0.78	0	0.69	0.97	0.3	0.8	0.5	17.123	
68	17	1	4	0	0	2	2	0	2	1	2	1.63	53		2.85</									

Statistical analysis **Adaptive (with Laser) vs Non-Adaptive (without Laser)** Following Experiment (ANOVA and Tukey):

*Table 42- Homogeneity of Variances Adaptive vs Non-Adaptive*

**Test of Homogeneity of Variances**

	Levene Statistic	df1	df2	Sig.
checkTotalLoss	.340	1	94	.561
percent_intervent	.422	1	94	.518
safety_intervension	1.121	1	94	.292
std_distance	.942	1	94	.334
false_depth_occlusion	1.002	1	94	.319
depth_occlusion	.010	1	94	.921

*Table 43- ANOVA Adaptive vs Non-Adaptive*

**ANOVA**

		Sum of Squares	df	Mean Square	F	Sig.
checkTotalLoss	Between Groups	3.650	1	3.650	13.395	.000
	Within Groups	25.618	94	.273		
	Total	29.268	95			
percent_intervent	Between Groups	4.348	1	4.348	32.897	.000
	Within Groups	12.423	94	.132		
	Total	16.771	95			
safety_intervension	Between Groups	7.042	1	7.042	12.588	.001
	Within Groups	52.583	94	.559		
	Total	59.625	95			
std_distance	Between Groups	.346	1	.346	4.266	.042
	Within Groups	7.616	94	.081		
	Total	7.962	95			
false_depth_occlusion	Between Groups	.085	1	.085	4.866	.030
	Within Groups	1.635	94	.017		
	Total	1.719	95			
depth_occlusion	Between Groups	.082	1	.082	5.132	.026
	Within Groups	1.496	94	.016		
	Total	1.577	95			

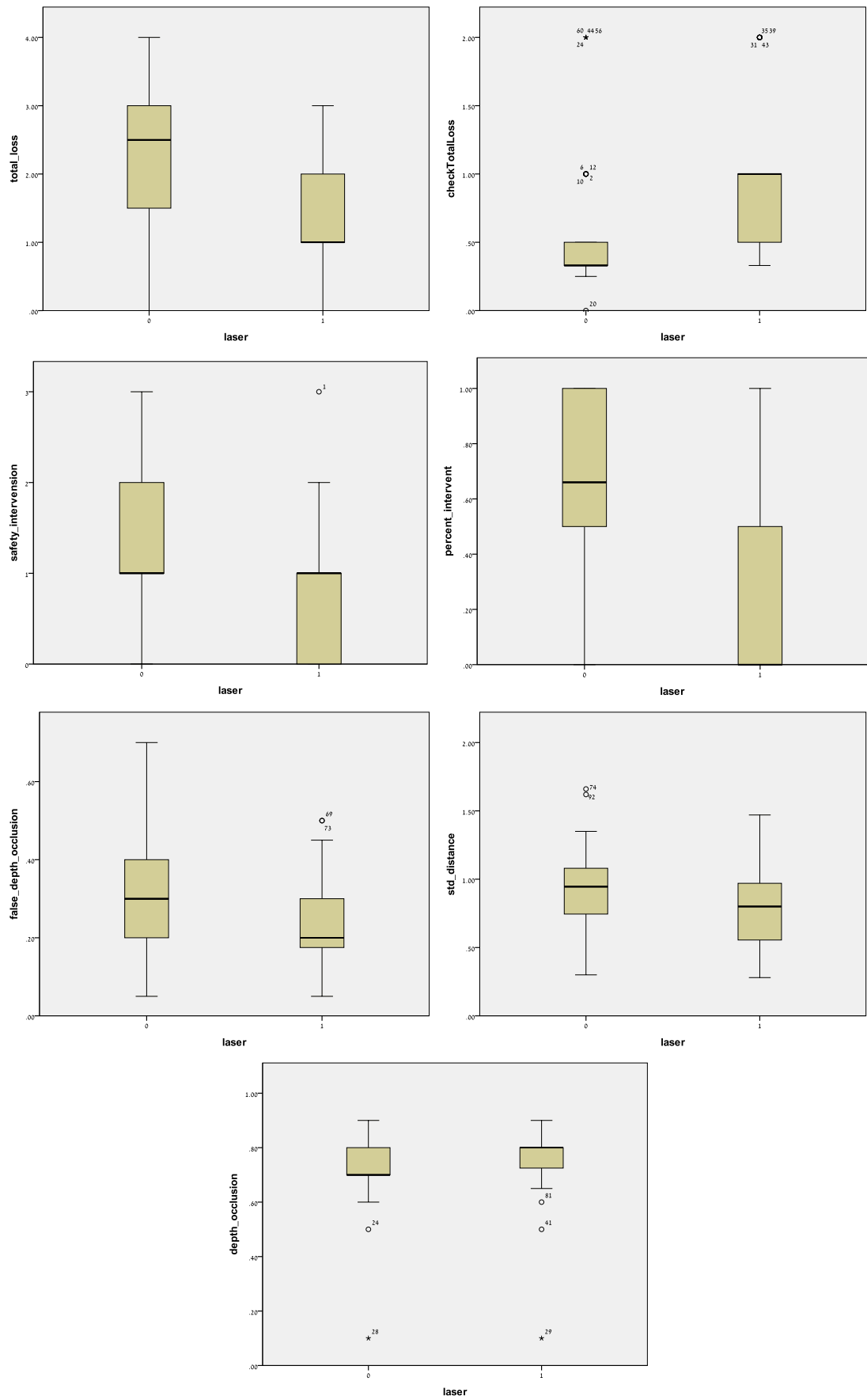


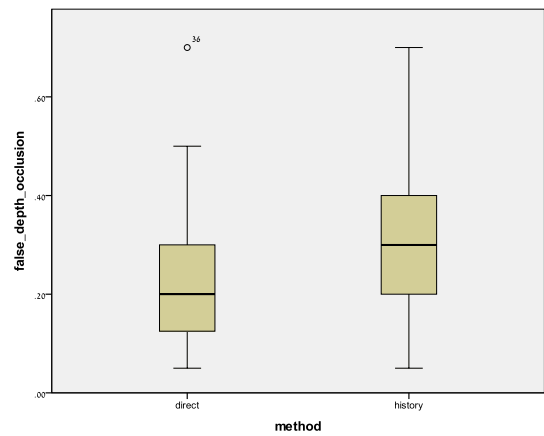
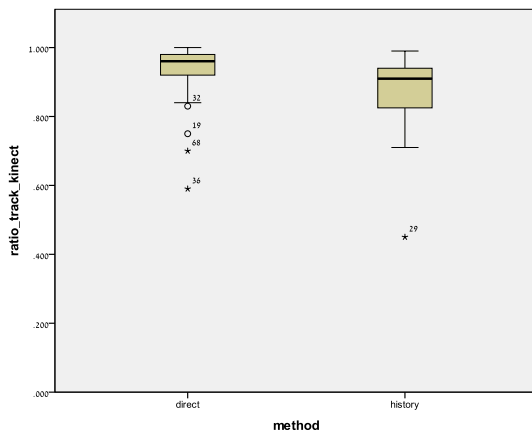
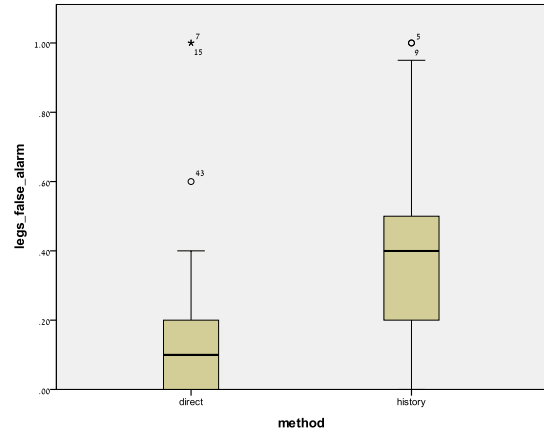
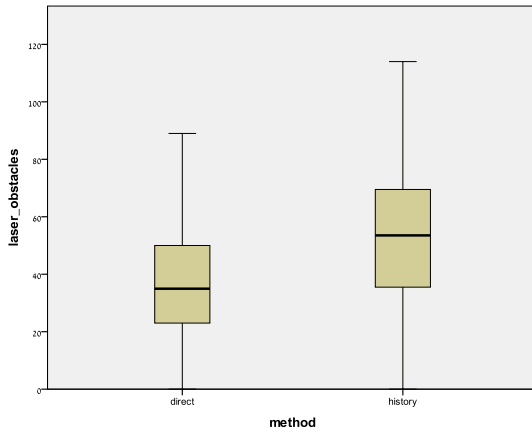
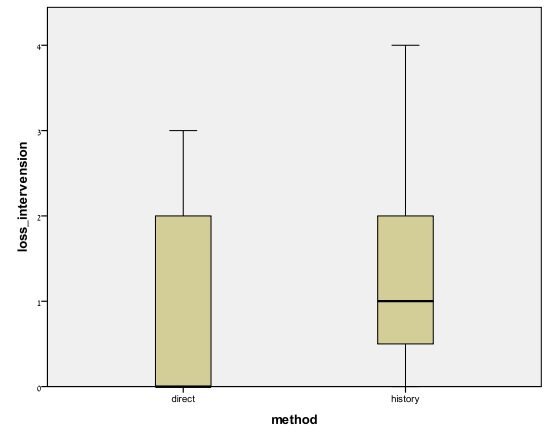
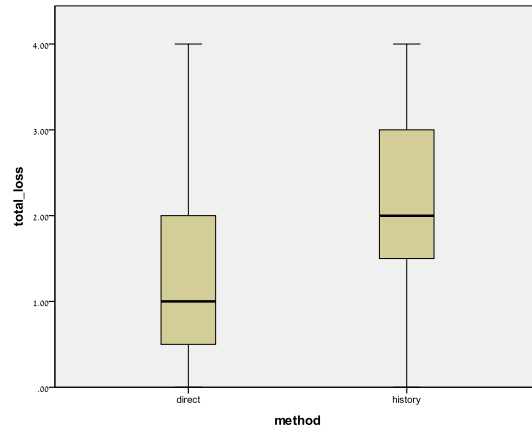
Figure 30- Descriptive Adaptive vs Non-Adaptive



Statistical analysis **Direct vs History** Following Experiment (ANOVA and Tukey):

*Table 44- ANOVA Direct vs History*

		ANOVA				
		Sum of Squares	df	Mean Square	F	Sig.
checkTotalLoss	Between Groups	3.650	1	3.650	13.395	.000
	Within Groups	25.618	94	.273		
	Total	29.268	95			
percent_intervent	Between Groups	4.348	1	4.348	32.897	.000
	Within Groups	12.423	94	.132		
	Total	16.771	95			
safety_intervension	Between Groups	7.042	1	7.042	12.588	.001
	Within Groups	52.583	94	.559		
	Total	59.625	95			
std_distance	Between Groups	.346	1	.346	4.266	.042
	Within Groups	7.616	94	.081		
	Total	7.962	95			
false_depth_occlusion	Between Groups	.085	1	.085	4.866	.030
	Within Groups	1.635	94	.017		
	Total	1.719	95			
depth_occlusion	Between Groups	.082	1	.082	5.132	.026
	Within Groups	1.496	94	.016		
	Total	1.577	95			



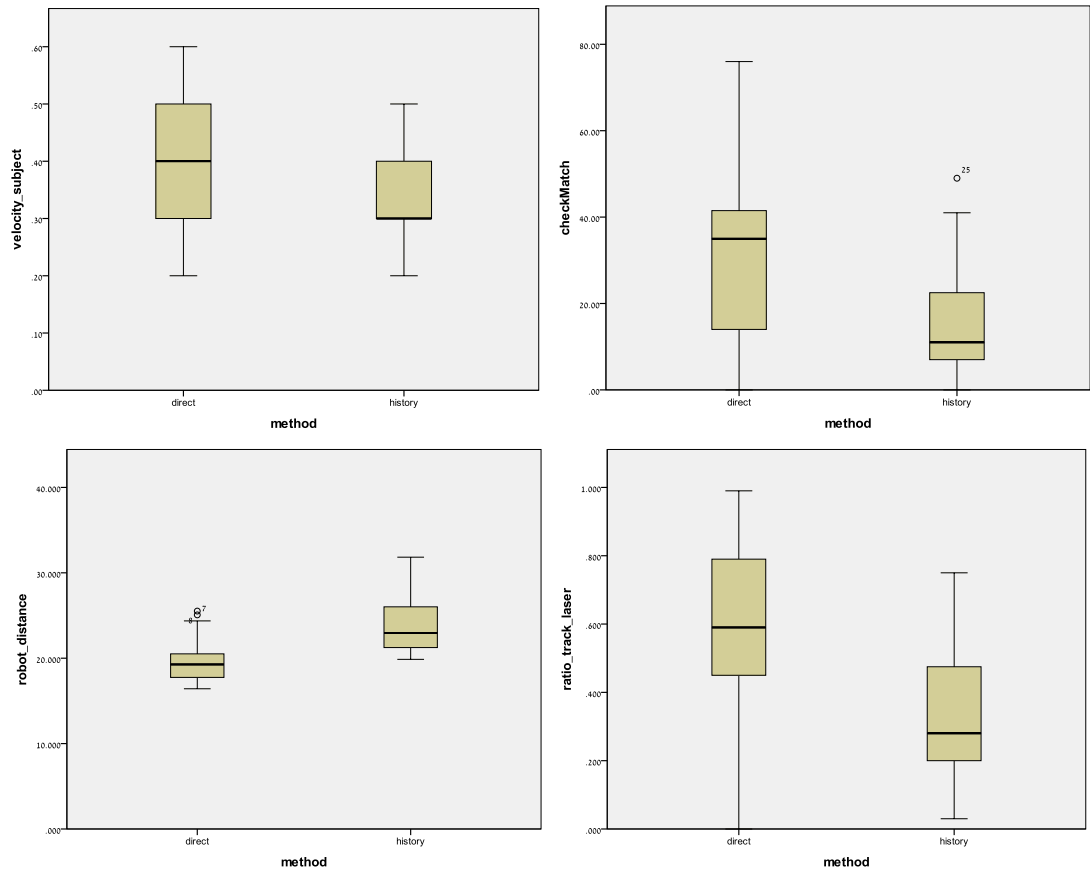


Figure 31- Descriptive Direct vs History

Statistical analysis **Gender (Males vs Females)** Following Experiment (ANOVA and Tukey):

Table 45- Homogeneity of Variances Gender (Males vs females)

Test of Homogeneity of Variances				
	Levene Statistic	df1	df2	Sig.
height	.015	1	94	.901
std_distance	.447	1	94	.505

Table 46- ANOVA Gender (Males vs females)

ANOVA						
		Sum of Squares	df	Mean Square	F	Sig.
height	Between Groups	.042	1	.042	10.950	.001
	Within Groups	.361	94	.004		
	Total	.403	95			
std_distance	Between Groups	.390	1	.390	4.836	.030
	Within Groups	7.572	94	.081		
	Total	7.962	95			

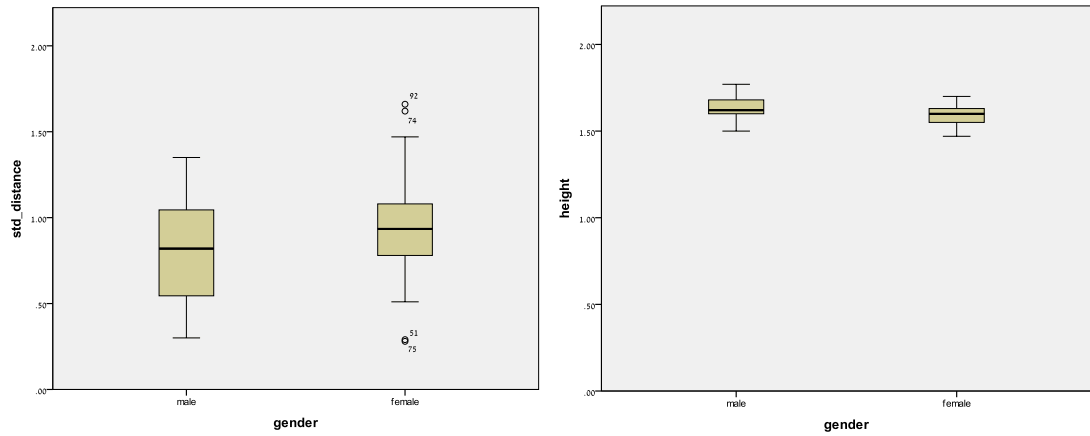


Figure 32- Descriptive Gender (Males vs females)

Statistical analysis **Subjects (24)** Following Experiment (ANOVA and Tukey):

Table 47- Homogeneity of Variances Subjects (24)

Test of Homogeneity of Variances			
height			
Levene Statistic	df1	df2	Sig.
1.456	23	72	.116

Table 48- ANOVA Subjects (24)

ANOVA					
height					
	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	.354	23	.015	22.775	.000
Within Groups	.049	72	.001		
Total	.403	95			

Table 49- Tukey HSD Subjects (24)

height

Tukey HSD<sup>a</sup>

subject	N	Subset for alpha = 0.05									
		1	2	3	4	5	6	7	8	9	10
4	4	1.5075									
16	4	1.5175									
18	4	1.5325	1.5325								
2	4	1.5575	1.5575	1.5575							
13	4	1.5600	1.5600	1.5600							
23	4	1.5700	1.5700	1.5700	1.5700						
20	4	1.5700	1.5700	1.5700	1.5700						
21	4		1.5950	1.5950	1.5950	1.5950					
15	4		1.5975	1.5975	1.5975	1.5975					
11	4			1.6050	1.6050	1.6050	1.6050				
10	4			1.6050	1.6050	1.6050	1.6050				
19	4			1.6125	1.6125	1.6125	1.6125	1.6125			
6	4			1.6150	1.6150	1.6150	1.6150	1.6150			
24	4			1.6175	1.6175	1.6175	1.6175	1.6175			
5	4			1.6250	1.6250	1.6250	1.6250	1.6250	1.6250		
14	4				1.6350	1.6350	1.6350	1.6350	1.6350		
12	4					1.6475	1.6475	1.6475	1.6475	1.6475	
8	4						1.6700	1.6700	1.6700	1.6700	
9	4						1.6700	1.6700	1.6700	1.6700	
17	4							1.6750	1.6750	1.6750	
1	4							1.6800	1.6800	1.6800	
3	4								1.6900	1.6900	1.6900
7	4									1.7125	1.7125
22	4										1.7525
Sig.		.136	.097	.068	.097	.415	.097	.068	.097	.097	.136

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 4.000.

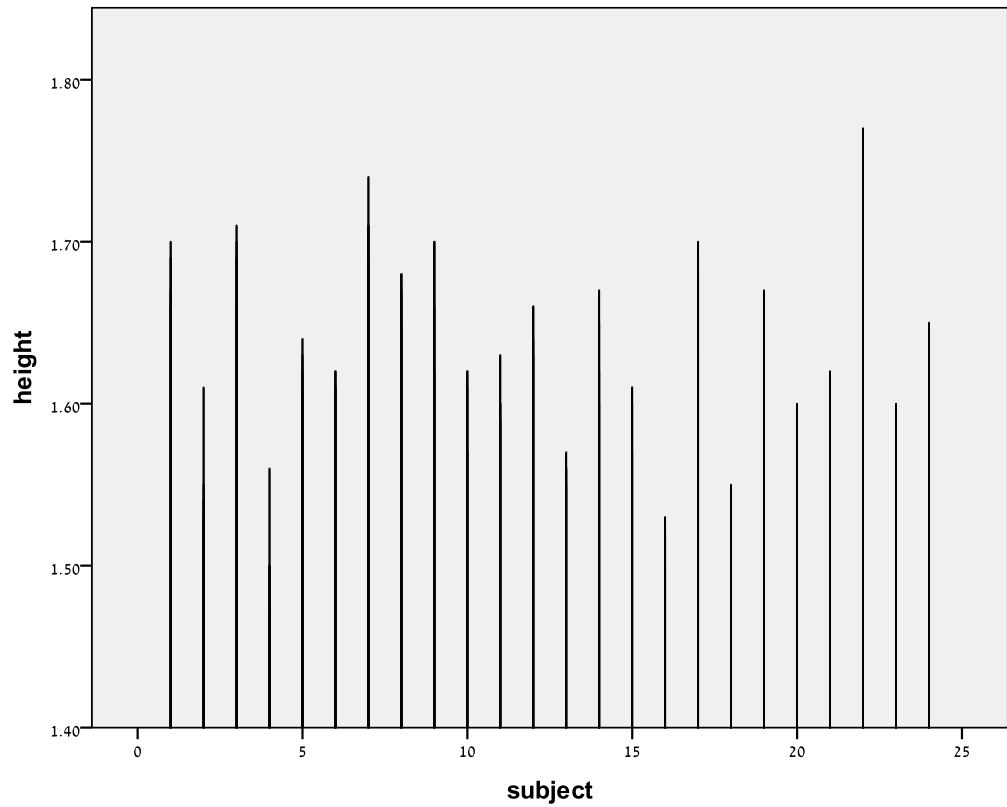


Figure 33- Descriptive Subjects (24)

Statistical analysis **Trials (4)** Following Experiment (ANOVA and Tukey):

Table 50- Homogeneity of Variances Trials (4)

Test of Homogeneity of Variances				
	Levene Statistic	df1	df2	Sig.
TotalLosses	2.506	3	92	.064
safety_intervension	.505	3	92	.680
average_distance	1.937	3	92	.129
ratio_track_kinect	2.366	3	92	.076
false_depth_occlusion	1.110	3	92	.349
depth_occlusion	1.335	3	92	.268
velocity_subject	.147	3	92	.931
RobotDistance	1.722	3	92	.168

Table 51- ANOVA Trials (4)

		ANOVA				
		Sum of Squares	df	Mean Square	F	Sig.
TotalLosses	Between Groups	40.865	3	13.622	19.392	.000
	Within Groups	64.625	92	.702		
	Total	105.490	95			
safety_intervension	Between Groups	8.125	3	2.708	4.838	.004
	Within Groups	51.500	92	.560		
	Total	59.625	95			
average_distance	Between Groups	10.517	3	3.506	16.211	.000
	Within Groups	19.894	92	.216		
	Total	30.410	95			
ratio_track_kinect	Between Groups	.085	3	.028	3.664	.015
	Within Groups	.716	92	.008		
	Total	.801	95			
false_depth_occlusion	Between Groups	.259	3	.086	5.430	.002
	Within Groups	1.461	92	.016		
	Total	1.719	95			
depth_occlusion	Between Groups	.235	3	.078	5.360	.002
	Within Groups	1.343	92	.015		
	Total	1.577	95			
velocity_subject	Between Groups	.114	3	.038	5.051	.003
	Within Groups	.694	92	.008		
	Total	.808	95			
RobotDistance	Between Groups	470.677	3	156.892	43.520	.000
	Within Groups	331.668	92	3.605		
	Total	802.345	95			

Table 52- Tukey HSD Trials (4)

Multiple Comparisons							
Tukey HSD							
Dependent Variable	(I) trial	(J) trial	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
						Lower Bound	Upper Bound
TotalLosses	1	2	-1.00000*	.24194	.000	-1.6331	-.3669
		3	.83333*	.24194	.005	.2003	1.4664
		4	-.20833	.24194	.825	-.8414	.4247
	2	1	1.00000*	.24194	.000	.3669	1.6331
		3	1.83333*	.24194	.000	1.2003	2.4664
		4	.79167*	.24194	.008	.1586	1.4247
	3	1	-.83333*	.24194	.005	-1.4664	-.2003
		2	-1.83333*	.24194	.000	-2.4664	-1.2003
		4	-1.04167*	.24194	.000	-1.6747	-.4086
	4	1	.20833	.24194	.825	-.4247	.8414
		2	-.79167*	.24194	.008	-1.4247	-.1586
		3	1.04167*	.24194	.000	.4086	1.6747
safety_intervension	1	2	-.583*	.216	.040	-1.15	-.02
		3	.167	.216	.867	-.40	.73
		4	-.333	.216	.416	-.90	.23
	2	1	.583*	.216	.040	.02	1.15
		3	.750*	.216	.004	.18	1.32
		4	.250	.216	.655	-.32	.82
	3	1	-.167	.216	.867	-.73	.40
		2	-.750*	.216	.004	-1.32	-.18
		4	-.500	.216	.102	-1.07	.07
	4	1	.333	.216	.416	-.23	.90
		2	-.250	.216	.655	-.82	.32
		3	.500	.216	.102	-.07	1.07
average_distance	1	2	-.16458	.13424	.612	-.5158	.1867
		3	.62917*	.13424	.000	.2779	.9804
		4	.49583*	.13424	.002	.1446	.8471
	2	1	.16458	.13424	.612	-.1867	.5158
		3	.79375*	.13424	.000	.4425	1.1450
		4	.66042*	.13424	.000	.3092	1.0117
	3	1	-.62917*	.13424	.000	-.9804	-.2779
		2	-.79375*	.13424	.000	-1.1450	-.4425
		4	-.13333	.13424	.754	-.4846	.2179
	4	1	-.49583*	.13424	.002	-.8471	-.1446
		2	-.66042*	.13424	.000	-1.0117	-.3092
		3	.13333	.13424	.754	-.2179	.4846
ratio_track_kinect	1	2	-.016208	.025459	.920	-.08282	.05041
		3	-.077792*	.025459	.015	-.14441	-.01118
		4	-.047375	.025459	.252	-.11399	.01924
	2	1	.016208	.025459	.920	-.05041	.08282
		3	-.061583	.025459	.081	-.12820	.00503
		4	-.031167	.025459	.613	-.09778	.03545
	3	1	.077792*	.025459	.015	.01118	.14441
		2	.061583	.025459	.081	-.00503	.12820
		4	.030417	.025459	.632	-.03620	.09703
	4	1	.047375	.025459	.252	-.01924	.11399
		2	.031167	.025459	.613	-.03545	.09778
		3	-.030417	.025459	.632	-.09703	.03620
false_depth_occlusion	1	2	-.04792	.03637	.554	-.1431	.0473
		3	.09583*	.03637	.048	.0007	.1910
		4	.02500	.03637	.902	-.0702	.1202
	2	1	.04792	.03637	.554	-.0473	.1431
		3	.14375*	.03637	.001	.0486	.2389
		4	.07292	.03637	.194	-.0223	.1681



	3	1	-.09583*	.03637	.048	-.1910	-.0007	
		2	-.14375*	.03637	.001	-.2389	-.0486	
		4	-.07083	.03637	.216	-.1660	.0243	
	4	1	-.02500	.03637	.902	-.1202	.0702	
		2	-.07292	.03637	.194	-.1681	.0223	
		3	.07083	.03637	.216	-.0243	.1660	
	depth_occlusion	1	2	-.02083	.03487	.933	-.1121	.0704
			3	-.08958	.03487	.056	-.1808	.0017
			4	.04792	.03487	.519	-.0433	.1392
		2	1	.02083	.03487	.933	-.0704	.1121
			3	-.06875	.03487	.206	-.1600	.0225
			4	.06875	.03487	.206	-.0225	.1600
3		1	.08958	.03487	.056	-.0017	.1808	
		2	.06875	.03487	.206	-.0225	.1600	
		4	.13750*	.03487	.001	.0462	.2288	
4		1	-.04792	.03487	.519	-.1392	.0433	
		2	-.06875	.03487	.206	-.1600	.0225	
		3	-.13750*	.03487	.001	-.2288	-.0462	
velocity_subject	1	2	-.01875	.02507	.877	-.0843	.0468	
		3	-.07708*	.02507	.014	-.1427	-.0115	
		4	-.07708*	.02507	.014	-.1427	-.0115	
	2	1	.01875	.02507	.877	-.0468	.0843	
		3	-.05833	.02507	.099	-.1239	.0073	
		4	-.05833	.02507	.099	-.1239	.0073	
	3	1	.07708*	.02507	.014	.0115	.1427	
		2	.05833	.02507	.099	-.0073	.1239	
		4	.00000	.02507	1.000	-.0656	.0656	
	4	1	.07708*	.02507	.014	.0115	.1427	
		2	.05833	.02507	.099	-.0073	.1239	
		3	.00000	.02507	1.000	-.0656	.0656	
RobotDistance	1	2	-.40421	.54811	.882	-1.8384	1.0300	
		3	4.80000*	.54811	.000	3.3658	6.2342	
		4	3.41433*	.54811	.000	1.9801	4.8485	
	2	1	.40421	.54811	.882	-1.0300	1.8384	
		3	5.20421*	.54811	.000	3.7700	6.6384	
		4	3.81854*	.54811	.000	2.3844	5.2527	
	3	1	-4.80000*	.54811	.000	-6.2342	-3.3658	
		2	-5.20421*	.54811	.000	-6.6384	-3.7700	
		4	-1.38567	.54811	.062	-2.8199	.0485	
	4	1	-3.41433*	.54811	.000	-4.8485	-1.9801	
		2	-3.81854*	.54811	.000	-5.2527	-2.3844	
		3	1.38567	.54811	.062	-.0485	2.8199	

\*. The mean difference is significant at the 0.05 level.

Table 53- Total\_Loss Trials (4)

TotalLosses				
Tukey HSD <sup>a</sup>				
trial	N	Subset for alpha = 0.05		
		1	2	3
3	24	.8333		
1	24		1.6667	
4	24		1.8750	
2	24			2.6667
Sig.		1.000	.825	1.000

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 24.000.

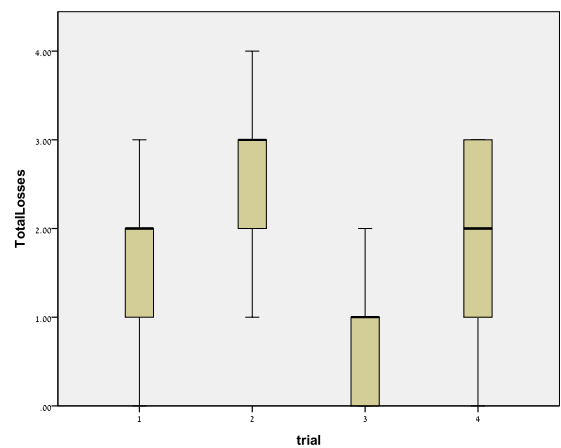


Table 54- Safety\_Intervention Trials (4)

safety_intervention			
Tukey HSD <sup>a</sup>			
trial	N	Subset for alpha = 0.05	
		1	2
3	24	.71	
1	24	.88	
4	24	1.21	1.21
2	24		1.46
Sig.		.102	.655

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 24.000.

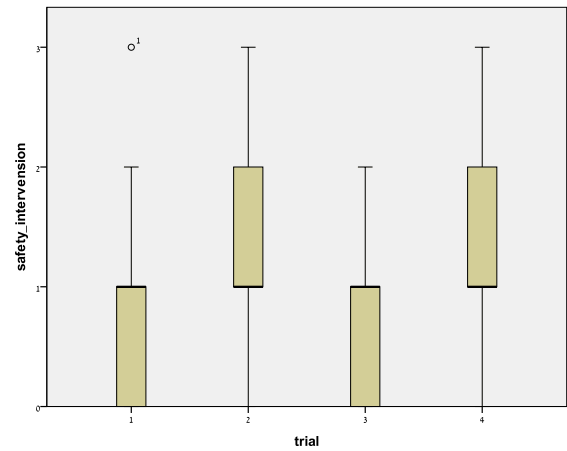


Table 55- Average\_Distance Trials (4)

average_distance			
Tukey HSD <sup>a</sup>			
trial	N	Subset for alpha = 0.05	
		1	2
3	24	2.9850	
4	24	3.1183	
1	24		3.6142
2	24		3.7788
Sig.		.754	.612

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 24.000.

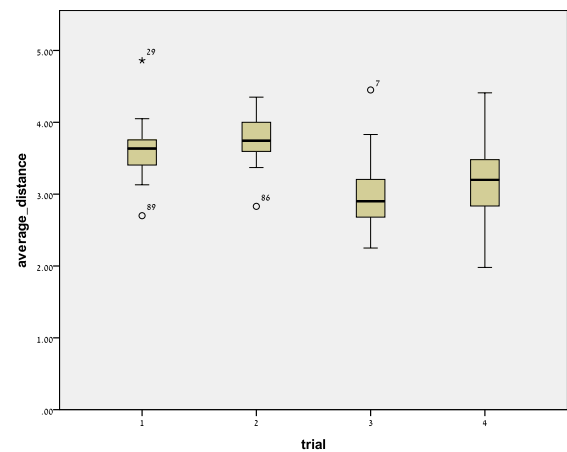


Table 56- Ratio\_Track\_Kinect Trials (4)

ratio_track_kinect			
Tukey HSD <sup>a</sup>			
trial	N	Subset for alpha = 0.05	
		1	2
1	24	.87221	
2	24	.88842	.88842
4	24	.91958	.91958
3	24		.95000
Sig.		.252	.081

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 24.000.

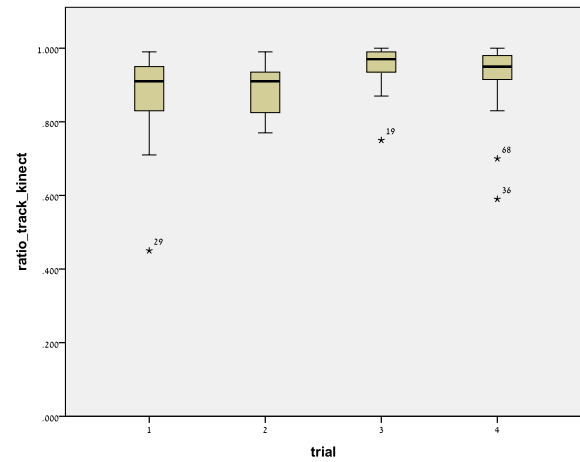


Table 57- False\_Depth\_Occlusion Trials (4)

false_depth_occlusion			
Tukey HSD <sup>a</sup>			
trial	N	Subset for alpha = 0.05	
		1	2
3	24	.1896	
4	24	.2604	.2604
1	24		.2854
2	24		.3333
Sig.		.216	.194

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 24.000.

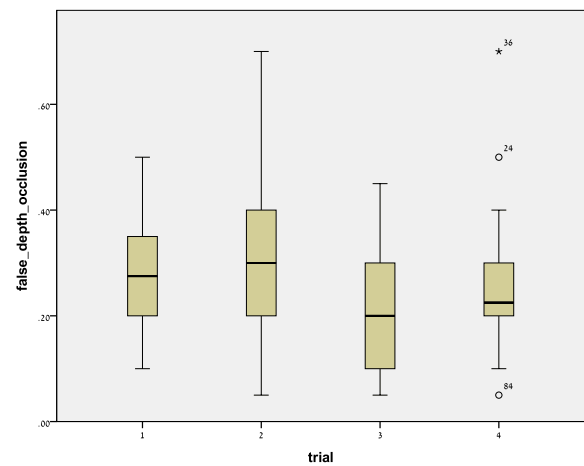


Table 58- Depth\_Occlusion Trials (4)

**depth\_occlusion**

Tukey HSD<sup>a</sup>

trial	N	Subset for alpha = 0.05	
		1	2
4	24	.6812	
1	24	.7292	.7292
2	24	.7500	.7500
3	24		.8188
Sig.		.206	.056

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 24.000.

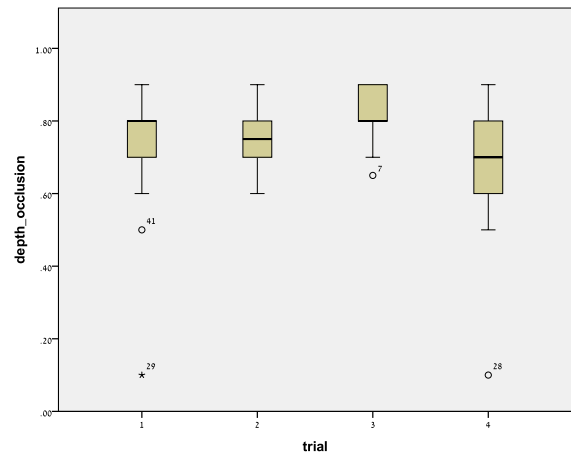


Table 59- Subject's\_Velocity Trials (4)

**velocity\_subject**

Tukey HSD<sup>a</sup>

trial	N	Subset for alpha = 0.05	
		1	2
1	24	.3271	
2	24	.3458	.3458
3	24		.4042
4	24		.4042
Sig.		.877	.099

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 24.000.

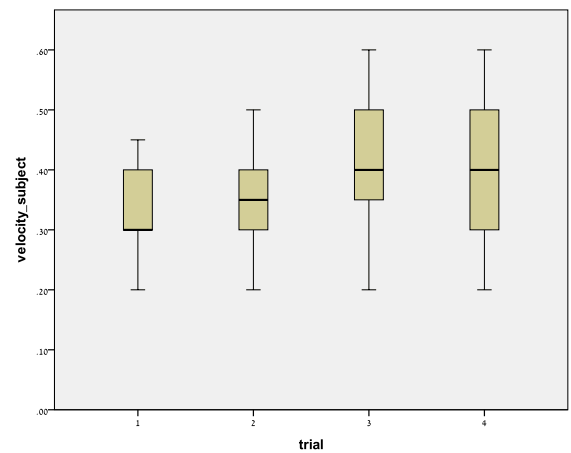


Table 60- Robot's\_Distance Trials (4)

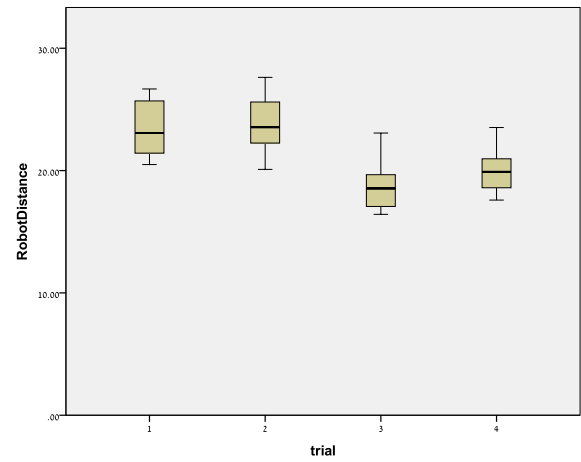
**RobotDistance**

Tukey HSD<sup>a</sup>

trial	N	Subset for alpha = 0.05	
		1	2
3	24	18.6055	23.4055 23.8097 .882
4	24	19.9912	
1	24		
2	24		
Sig.		.062	

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 24.000.



## Appendix D- C++ codes

### 1. Depth Occlusions Detection

```
#include <stdio.h> #include <stdlib.h> #include "ros/ros.h" #include "math.h"
#include "std_msgs/String.h" #include "std_msgs/Float32.h" #include "nav_msgs/Odometry.h"
#include "geometry_msgs/Twist.h" #include "sensor_msgs/LaserScan.h"
#include "opt_msgs/TrackArray.h" #include <ros/console.h>
#include <cv_bridge/cv_bridge.h> #include <sensor_msgs/image_encodings.h>
#include "opt_msgs/DetectionArray.h" #include <occlusions/sideOcclusions.h>
#include <opencv2/imgproc/imgproc.hpp> #include <opencv2/highgui/highgui.hpp>

occlusions::sideOcclusions bool_msg; //6 boolians variables
double AgeThreshold=0; //how "old" is the ID
double ConfidenceTheshold=1.1; //from the SVM+HOG classifier- confidence for a real person
double HeightTheshold=1.4; //height in meter of the person (minimum)
double HeightMaxTheshold=2.0; //height in meter of the person (maximum)
namespace enc = sensor_msgs::image_encodings;
static const std::string OPENCV_WINDOW = "Image window";
class ImageConverter
{
    ros::NodeHandle n;
    image_transport::ImageTransport it_;
    image_transport::Subscriber image_sub;
    image_transport::Publisher image_pub;
    ros::Subscriber person_sub = n.subscribe("/tracker/tracks", 10,
    &ImageConverter::boxCallback, this); //get the track parameters
    ros::Publisher side=
    (n.advertise<occlusions::sideOcclusions>("occlusions/sideOcclusions",10)); //the 6 boolians
    double xmin=0; //left side of the BBC (Bounding Box Coordinates)
    double ymin=0; //top side of the BBC
    double xmax=0; //right side of the BBC
    double ymax=0; //bottom side of the BBC
    double distance; //from Open_PTrack trackers- distance in meters to the detect person
    double confidence; //from Open_PTrack- the SVM+HOG classifier- confidence for a real person
    double age; //from Open_PTrack trackers
    double height; //from Open_PTrack trackers
    double xc; //center of the BBC
    float depth; //pixel depth value at xc,y
    float personDepth; //distance*1000
    double depthTheshold=3.0; //threshold for detect closer pixels from the personDepth
    bool validTrack; //good track
    int nbOfTracks; //number of ID tracks
    float normalize; //normalize the depth value

public:
    ImageConverter()
    : it_(n){
        image_sub = it_.subscribe("/kinect2_head/depth_rect/image", 10,
    &ImageConverter::imageCallback, this); //depth image (same as Open_PTrack uses)
        image_pub = it_.advertise("/image_converter/output_video", 1);
        cv::namedWindow(OPENCV_WINDOW);}
    ~ImageConverter()
    {cv::destroyWindow(OPENCV_WINDOW);}

    void boxCallback(const opt_msgs::TrackArray::ConstPtr& msg){ //get all the tracks parameters
        validTrack=false;
        nbOfTracks=msg->tracks.size();
        if (nbOfTracks>0) {
            for(int i=0;i<nbOfTracks && !validTrack;i++){
                //oldest track which is older than the age threshold, above the confidence
                threshold, above the height threshold and under max height threshold
                if ((msg->tracks[i].age>AgeThreshold) && (msg->tracks[i].confidence>ConfidenceTheshold) && (msg->tracks[i].height>HeightTheshold) && (msg->tracks[i].height<HeightMaxTheshold)){
                    xmin=msg->tracks[i].box_2D.x; //left side of the BBC (Bounding Box Coordinates)
                    ymin=msg->tracks[i].box_2D.y; //top side of the BBC
                    xmax=xmin+msg->tracks[i].box_2D.width; //right side of the BBC
                    ymax=ymin+msg->tracks[i].box_2D.height; //bottom side of the BBC
                    distance=msg->tracks[i].distance; //Open_PTrack-distance in meters to a person
                    confidence=msg->tracks[i].confidence; //Open_PTrack- from the SVM+HOG classifier
                    height=msg->tracks[i].height; //from Open_PTrack trackers
                    age=msg->tracks[i].age; //from Open_PTrack trackers
                    validTrack=true;
                }
            }
        }
    }

    void imageCallback(const sensor_msgs::ImageConstPtr& msg) //working on the depth image
```

```

{cv_bridge::CvImagePtr cv_ptr;
try
{cv_ptr = cv_bridge::toCvCopy(msg, sensor_msgs::image_encodings::TYPE_16UC1);} //now
cv_ptr is the matrix
catch (cv_bridge::Exception& e)
{ROS_ERROR("cv_bridge exception: %s", e.what());
return;}
image_pub.publish(cv_ptr->toImageMsg()); // Output modified video stream
xc=(xmin+xmax)/2; //center of the BBC
personDepth=distance*1000; //to avoid an error from calculate the depth only
from one pixel, it's better to calculate from the all distance from the person and multipile by
1000 to get milimeters
depth=personDepth*255/pow(2,16); //to normalize to 255
//left and right detections
int downCut= round((ymax-ymin)/8); //cut lower part of a person to reduce floor alarm
int smallOcclusions= round(((xc-xmin)/3)*(ymax-ymin)*7/8); //detect small occlusion
int bigOcclusions= round(((xc-xmin)/2)*(ymax-ymin)*7/8); //detect big occlusion
int marginAdd= round(10/distance); //add margin depend on distance
int countLeft= 0;
int countRight= 0;
bool smallLeftOcclusions= false; //detect occlusion from the left side of the robot
bool bigLeftOcclusions= false; //detect occlusion from the left side of the robot
bool LeftWall= false; //detect a "tall" occlusion from the left side of the
robot, like a wall for all the y axis of the person bounding box
int countLeftWall= 0;
bool smallRightOcclusions= false; //detect occlusion from the right side of the robot
bool bigRightOcclusions= false; //detect occlusion from the right side of the robot
bool RightWall= false; //detect a "tall" occlusion from the right side of the
robot, like a wall for all the y axis of the person bounding box
int countRightWall= 0;
//left side
for (short int i=xmin-marginAdd;i<xc-5;i++){ //over each colom from the left with
margin up to the center minus 5
countLeftWall= 0;
for (short int j=ymin;j<ymax-downCut;j++){ //over all the specific colom from up
to down without the lower part to avoid the floor
normalize=cv_ptr->image.at<short int>(cv::Point(i,j)); //get the pixel depth value
normalize=normalize*255/pow(2,16); //normalize the depth value to 0-255
if (normalize<(depth-depthTheshold)){ //closer than the personDepth
countLeft++;
countLeftWall++;
if (countLeftWall==ymax-downCut-ymin){LeftWall=true;} //if the all colom is
closer than this is a wall
}
}
}
//right
for (short int i=xc+5;i<xmax+marginAdd;i++){ //over each colom from the center plus 5
up to the right with margin
countRightWall= 0;
for (short int j=ymin;j<ymax-downCut;j++){ //over all the specific colom from up to
down without the lower part to avoid the floor
normalize=cv_ptr->image.at<short int>(cv::Point(i,j)); //get the pixel depth value
normalize=normalize*255/pow(2,16); //normalize the depth value to 0-255
if (normalize<(depth-depthTheshold)){ //closer than the personDepth
countRight++;
countRightWall++;
if (countRightWall==ymax-downCut-ymin){RightWall=true;} //if the all colom is
closer than this is a wall
}
}
}
if (countLeft>smallOcclusions&&countLeft<bigOcclusions){ //if number of pixels from the
left are between smallLeftOcclusions and bigLeftOcclusions this is a smallLeft
smallLeftOcclusions=true;}
else if (countLeft>bigOcclusions){ //if number of pixels from the left are more than
bigLeftOcclusions this is a bigLeft
bigLeftOcclusions=true;}
if (countRight>smallOcclusions&&countRight<bigOcclusions){ //if number of pixels from
the right are between smallRightOcclusions and bigRightOcclusions this is a smallRight
smallRightOcclusions=true;}
else if (countRight>bigOcclusions){ //if number of pixels from the right are more
than bigRightOcclusions this is a bigRight
bigRightOcclusions=true;}
//publish the boolians variables
bool_msg.bigLeft=bigLeftOcclusions;
bool_msg.smallLeft=smallLeftOcclusions;
bool_msg.wallLeft=LeftWall;
bool_msg.bigRight=bigRightOcclusions;

```

```

        bool_msg.smallRight=smallRightOcclusions;
        bool_msg.wallRight=RightWall;
        side.publish(bool_msg);
    }
};

int main(int argc, char **argv){
    ros::init(argc, argv, "image_converter");
    ImageConverter ic;
    ros::NodeHandle n;
    ros::spin();
    return 0;
}

```

## 2. Vision Occlusions Detection

```

#include <stdio.h> #include <stdlib.h> #include "ros/ros.h" #include "math.h"
#include "std_msgs/String.h" #include "std_msgs/Float32.h" #include "nav_msgs/Odometry.h"
#include "geometry_msgs/Twist.h" #include "sensor_msgs/LaserScan.h"
#include <ros/console.h> #include <image_transport/image_transport.h>
#include <cv_bridge/cv_bridge.h> #include <sensor_msgs/image_encodings.h>
#include "opt_msgs/DetectionArray.h" #include <occlusions/sideOcclusions.h>
#include <opencv2/imgproc/imgproc.hpp> #include <opencv2/highgui/highgui.hpp>
#define PI 3.14159265
occlusions::sideOcclusions bool_msg; //6 boolians variables
double AgeThreshold=0; //how "old" is the ID
double ConfidenceTheshold=1.1; //from the SVM+HOG classifier- confidence for a real person
double HeightTheshold=1.4; //height in meter of the person (minimum)
double HeightMaxTheshold=2.0; //height in meter of the person (maximum)
namespace enc = sensor_msgs::image_encodings;
using namespace cv; using namespace std;
static const std::string OPENCV_WINDOW = "Image window";

class ImageConverter
{
    ros::NodeHandle n;
    image_transport::ImageTransport it_;
    image_transport::Subscriber image_sub;
    image_transport::Publisher image_pub;
    ros::Subscriber person_sub = n.subscribe("/tracker/tracks", 10,
    &ImageConverter::boxCallback, this); //get the track parameters
    ros::Publisher side=
    (n.advertise<occlusions::sideOcclusions>("occlusions/sideOcclusions",10)); // the 6 boolians
    double xmin=0; //left side of the BBC from the DEPTH image
    double ymin=0; //top side of the BBC from the DEPTH image
    double xmax=0; //right side of the BBC from the DEPTH image
    double ymax=0; //bottom side of the BBC from the DEPTH image
    double xcenter; //the center of the box in x axis at the DEPTH image
    double distance; //Open_PTrack- distance in meters to the detect person
    double confidence; //Open_PTrack- from the SVM+HOG classifier
    double age; //from Open_PTrack trackers
    double height; //from Open_PTrack trackers
    double rgbxmin=0; //left side of the BBC from the MONO image
    double rgbymax=0; //top side of the BBC from the MONO image
    double rgbxmax=0; //right side of the BBC from the MONO image
    double rgbymax=0; //bottom side of the BBC from the MONO image
    double xcBox; //the center of the box in the ROI
    bool validTrack; //good track
    int nbOfTracks; //number of ID tracks
    bool LeftWall= false; //detect left wall
    bool RightWall= false; //detect right wall

public:
    ImageConverter()
    : it_(n)
    {image_sub = it_.subscribe("/kinect2_head/mono_rect/image", 10,
    &ImageConverter::imageCallback, this); //get the mono_rect image (gray image)
    image_pub = it_.advertise("/image_converter/output_video", 1);
    cv::namedWindow(OPENCV_WINDOW);}
    ~ImageConverter()
    {cv::destroyWindow(OPENCV_WINDOW);}

    void boxCallback(const opt_msgs::TrackArray::ConstPtr& msg){//get all the tracks parameters
        validTrack=false;
        nbOfTracks=msg->tracks.size();
        if (nbOfTracks>0) {

```



```

        for(int i=0;i<nbOfTracks && !validTrack;i++){
//oldest track which is older than the age threshold, above the confidence threshold, above the
height threshold and under max height threshold
            if ((msg->tracks[i].age>AgeThreshold) && (msg-
>tracks[i].confidence>ConfidenceThreshold) && (msg->tracks[i].height>HeightThreshold) && (msg-
>tracks[i].height<HeightMaxThreshold)){
                xmin=msg->tracks[i].box_2D.x;           //left side of the BBC from the DEPTH image
                ymin=msg->tracks[i].box_2D.y;           //top side of the BBC from the DEPTH image
                xmax=xmin+msg->tracks[i].box_2D.width;   //right side of the BBC from the DEPTH image
                ymax=ymin+msg->tracks[i].box_2D.height;  //bottom side of the BBC from the DEPTH image
                distance=msg->tracks[i].distance;        //Open_PTrack- distance in meters to a person
                confidence=msg->tracks[i].confidence;    //Open_PTrack- from the SVM+HOG classifier
                height=msg->tracks[i].height;           //from Open_PTrack trackers
                age=msg->tracks[i].age;                 //from Open_PTrack trackers
                validTrack=true;
            }
        }
    }
}

void imageCallback(const sensor_msgs::ImageConstPtr& msg) //working on the depth image
{cv_bridge::CvImagePtr cv_ptr;
try
    {cv_ptr = cv_bridge::toCvCopy(msg);} //now cv_ptr is the matrix of the image
catch (cv_bridge::Exception& e)
    {ROS_ERROR("cv_bridge exception: %s", e.what());
return;}
cv::Mat deleteMargin= cv::Mat::zeros(1080,1710,0);
deleteMargin = cv_ptr->image(Rect(104,0,1710,1080)).clone(); //because a different FOV
between depth and RGB i delete the 105 pixels from each side to reduce the FOV different
cv::Size size(960,540); //size of the depth image
cv::resize(deleteMargin,deleteMargin,size); //resize the gray mono image to the depth image
size because the resolution of mono_rect is twice the resolution of depth_rect
xcenter=(xmin+xmax)/2; //the center of the box in x axis at the DEPTH image
rgbxmin=xmin*2+round(((270-xcenter)/3)-distance*2); //left side of the BBC from the MONO
image with react to the center of the image and distance because different FOV
rgbxmax=rgbxmin+(xmax-xmin)*1.4; //right side of the BBC from the MONO image with
more width to cover the all person
rgbymin=ymin; //top side of the BBC from the MONO image
rgbymax=rgbymin+(ymax-ymin)*1.3; //bottom side of the BBC from the
MONO image with more for the legs but not the ground
int marginAdd= round(50/distance); //add margin depend on distance
LeftWall= false; //detect a "tall" vertical occlusion from the left side of the robot
RightWall= false; //detect a "tall" vertical occlusion from the right side of the robot
xcBox=((xmax-xmin)*1.4+marginAdd*2)/2; //the center of the box in the ROI
cv::Mat temp= cv::Mat::zeros(540,960,0); //temp with the maximum size of the MONO image
int top = (int) (0.01*temp.rows); //for add borders to the image
int bottom = (int) (0.01*temp.rows);
int left = (int) (0.01*temp.cols);
int right = (int) (0.01*temp.cols);
if(rgbxmin-marginAdd*2 >= 0 && rgbymin >= 0 && rgbxmax+marginAdd < deleteMargin.cols && rgbymax
< deleteMargin.rows && (xmax-xmin)*1.4+marginAdd>0 && (ymax-ymin)*1.3>0){
    temp = deleteMargin(Rect(rgbxmin-marginAdd,rgbymin,(xmax-xmin)*1.4+marginAdd,(ymax-
ymin)*1.3)).clone();} //take only the BBC with the person with margin depend on distance
cv::GaussianBlur(temp, temp, cv::Size(3,3), 0); //gaussian blur 3*3
cv::Canny(temp, temp, 50.0, 300.0, 3, false); //canny edge detector
cv::Mat element1= cv::getStructuringElement(cv::MORPH_RECT, cv::Size(5,5), cv::Point(-1,-
1)); // 5*5 open element
cv::Mat element2= cv::getStructuringElement(cv::MORPH_RECT, cv::Size(5,5), cv::Point(-1,-
1)); // 5*5 close element
cv::dilate(temp, temp, element1); //open the pixels
cv::erode(temp, temp, element2); //close the pixels
cv::Mat temp2=temp;
cv::cvtColor(temp2, temp2, cv::COLOR_GRAY2BGR); //color mat for show
copyMakeBorder( temp2, temp2, top, bottom, left, right, BORDER_CONSTANT,
cv::Scalar(255,255,255) ); //add white borders to help detect contours
std::vector<std::vector<cv::Point> > contours; //vector of vector points- contours
cv::findContours(temp,contours,CV_RETR_TREE,CV_CHAIN_APPROX_SIMPLE); //find contours
for (int i=0; i<contours.size(); i++){
    if(contours[i].size()>(rgbymax-rgbymin)*1.5){ //only if the contour is bigger
than the whole height of the ROI multiple 1.5
        cv::drawContours(temp2,contours,i,cv::Scalar(0,255,0),8,8);} //draw green
    else {contours.pop_back();} //delete small contours from vector "contours"
}
cv::vector<Vec4i> lines; //"lines" contain 4 values for xStart,yStart,xEnd,yEnd
HoughLinesP(temp, lines, 1, CV_PI / 180, 50, (rgbymax-rgbymin)*0.5, 0 ); //find
straight lines with Hough that are bigger than the half height of the ROI
for (size_t i = 0; i < lines.size(); i++)
    {cv::Vec4i l = lines[i];

```

```

        if (abs(l[0]-l[2])<(rgbxmax-rgbxmin)/10) { //for only vertical lines depend on the
width of the person box divide by 10 (only 1/10 size of the width)
            line(temp2, Point(l[0], l[1]), Point(l[2], l[3]), Scalar(0, 0, 255), 10, 4);
//draw red line
            //left
            if (((l[0]>0) && (l[0]<xcBox-5)) || ((l[2]>0) && (l[2]<xcBox-5))){ //if the
edges of the straight line is inside the ROI from the left to the center minus 5
                LeftWall=true;
            }
            //right
            if (((l[0]>xcBox+5) && (l[0]<temp.cols)) || ((l[2]>xcBox+5) &&
(l[2]<temp.cols))){ //if the edges of the straight line is inside the ROI from the center plus
5 to the right
                RightWall=true;
            }
            else {lines.pop_back();} //delete all the other lines from the vector "lines"
        }
    }
    rectangle(temp2,cv::Point(rgbxmin,rgbymin),cv::Point(rgbxmax,rgbymax),cv::Scalar(0,255,0), 10);
    cv::imshow(OPENCV_WINDOW, temp2); // Update GUI Window
    cv::waitKey(3);
    image_pub.publish(cv_ptr->toImageMsg()); // Output modified video stream
    bool_msg.wallLeft=LeftWall;
    bool_msg.wallRight=RightWall;
    side.publish(bool_msg);
}
};

int main(int argc, char **argv){
    ros::init(argc, argv, "image_converter");
    ImageConverter ic;
    ros::NodeHandle n;
    ros::spin();
    return 0;
}

```

### 3. Combination of Depth and Vision Occlusions Detection

Union of Depth and Vision Occlusion Detection codes according to the distance of the person.

```

    double changeDepthToRGB=5.0; //which distance to change between depth to RGB (meters)
    double distance; //Open_PTrack- distance in meters to the detect person
    if(distance<changeDepthToRGB){ //all the code from Depth Occlusion Detection }
    if(distance>changeDepthToRGB){ //all the code from Vision Occlusion Detection }

```

### 4. Obstacles Avoidance

```

#include <stdio.h> #include <stdlib.h> #include "ros/ros.h" #include "std_msgs/String.h"
#include "std_msgs/Float64.h" #include "std_msgs/Float32.h" #include "std_msgs/Bool.h"
#include "geometry_msgs/Twist.h" #include "geometry_msgs/Point32.h"
#include "sensor_msgs/PointCloud.h" #include <tf/transform_listener.h>
#include "pcl_ros/point_cloud.h" #include "pcl_ros/transforms.h"
#include <obstacles/laserObstacles.h> #include <people_msgs/PositionMeasurementArray.h>
#include "opt_msgs/TrackArray.h"
obstacles::laserObstacles ob_msg; //publise two Booleans and velocity command

class LaserObstacles
{
    ros::NodeHandle n;
    tf::TransformListener tf_listener;
    ros::Subscriber cmdVel = n.subscribe("/cmd_vel", 10, &LaserObstacles::velocityCallback,
this);
    ros::Subscriber sub_laser = n.subscribe("/RosAria/S3Series_1_pointcloud", 10,
&LaserObstacles::LaserCallback,this); //get the laser point
    ros::Subscriber sub_people= n.subscribe("/people_tracker_measurements", 10,
&LaserObstacles::LaserLegsCallback, this); //get the leg detector point of the person
    ros::Subscriber sub_people_kinect= n.subscribe("/tracker/tracks", 10,
&LaserObstacles::KinectCallback, this); //get the kinect point of the person
    ros::Subscriber sub2= n.subscribe("/Pan_Feedback", 10, &LaserObstacles::panCallback, this);
    ros::Subscriber sub3= n.subscribe("/Pan_Error_Command", 10,
&LaserObstacles::smallErrorCallback, this);
    ros::Publisher pub=(n.advertise<obstacles::laserObstacles>
("/obstacles/laserObstacles",10));
    double KpDistanceCheck=3; //Kp for distance depend on linear velocity

```

```

double DistanceCheck=0.8; // minimum distance in front of the robot
double WidthCheck= 0.5; //for each side
double DistanceSlowDownCheck= 1.5; //distance from an obstacle to slow down
double angularVelocity; //angular velocity of the robot
double linearVelocity; //linear velocity of the robot
double xLaserPerson; double yLaserPerson; double xKinectPerson; double yKinectPerson;
double radiusPerson=1.0; //radius around a person that clear from obstacles
double AngleErrorPan=0; //the angle of the Pan related to the center of the robot
bool smallError=false; //declare a small error to avoid small movements of the Pan
double smallErrorThreshold=0.01; //threshold for avoid small movements of the Pan
double AngleSmallError=0; //the angle of the person related to the center of the kinect

void smallErrorCallback(const std_msgs::Float32::ConstPtr& msg)
{
    AngleSmallError=msg->data;
    if ((abs(AngleSmallError)<smallErrorThreshold)&&
(abs(AngleErrorPan)<0.01)){smallError=true;}
    else {smallError=false;}}

void panCallback(const std_msgs::Float32::ConstPtr& msg)
{
    AngleErrorPan=msg->data;}

void LaserLegsCallback(const people_msgs::PositionMeasurementArray::ConstPtr& msg)
{
    int nbOfTracksLaser=msg->people.size();
    if (nbOfTracksLaser>0) { //Extract coordinates of first detected person from laser
        for(int i=0; i<nbOfTracksLaser;i++){
            xLaserPerson=msg->people[i].pos.x; yLaserPerson=msg->people[i].pos.y;}}
    else{ xLaserPerson=xKinectPerson; yLaserPerson=yKinectPerson;}}

void KinectCallback(const opt_msgs::TrackArray::ConstPtr& msg)
{
    int nbOfTracksKinect=msg->tracks.size();
    if (nbOfTracksKinect>0) { //Extract coordinates of first detected person from Kinect
        for(int i=0; i<nbOfTracksKinect;i++){
            xKinectPerson=((msg->tracks[i].distance)*cos(AngleSmallError+AngleErrorPan));
            yKinectPerson=((msg->tracks[i].distance)*sin(AngleSmallError+AngleErrorPan));}}}

void LaserCallback(const sensor_msgs::PointCloud::ConstPtr& msg)
{
    sensor_msgs::PointCloud pc_out;
    bool obstacle=false; //true if an obstacle was found
    bool SlowDown=false; //true if an obstacle was found in slowdown distance
    double angularCommand; double linearCommand; //velocity to avoid obstacle
    double XclosestObstacle=100.0; double YclosestObstacle=100.0;
    double DclosestObstacle=100.0; //X, Y, distance- 100 when there is no obstacle
    tf_listener.waitForTransform("/laser_frame", (*msg).header.frame_id, (*msg).header.stamp,
ros::Duration(5.0));
    tf_listener.transformPointCloud("/laser_frame", *msg, pc_out);
    for (int i=0; i<pc_out.points.size(); i++)
    {
        if (((pc_out.points[i].x < KpDistanceCheck*linearVelocity) || (pc_out.points[i].x <
DistanceCheck)) && (pc_out.points[i].x >-abs(angularVelocity)) &&
        ( ((sqrt(pow(pc_out.points[i].x-xLaserPerson,2)+pow(pc_out.points[i].y-
yLaserPerson,2))>radiusPerson) && (xLaserPerson!=0.0))||
        ((sqrt(pow(pc_out.points[i].x-xKinectPerson,2)+pow(pc_out.points[i].y-
yKinectPerson,2))>radiusPerson) && (xKinectPerson!=0.0)) )) //an obstacle inside the
DistanceCheck and more than radiusPerson from a detected legs or detected person from the Kinect
        {
            if (((pc_out.points[i].y < WidthCheck*(1+angularVelocity)) && (pc_out.points[i].y > -
WidthCheck*(1-angularVelocity)))){ //2 conditions: 1. y smaller than positive WidthCheck
                multiply the power of the turn of the robot; 2.y bigger than negative WidthCheck multiply the
                power of the turn of the robot;
                double pointDistance =sqrt(pow(pc_out.points[i].x,2)+pow(pc_out.points[i].y,2));
                //to get the closet obstacle
                if (pointDistance<DclosestObstacle){ XclosestObstacle=pc_out.points[i].x;
                    YclosestObstacle=pc_out.points[i].y; DclosestObstacle=pointDistance;}
                obstacle=true;}
        }
    }

    if (((XclosestObstacle < KpDistanceCheck*linearVelocity) || (XclosestObstacle <
DistanceCheck)) && (XclosestObstacle >-abs(angularVelocity)) &&
        ( ((sqrt(pow(XclosestObstacle-xLaserPerson,2)+pow(YclosestObstacle-
yLaserPerson,2))>radiusPerson) && (xLaserPerson!=0.0))||
        ((sqrt(pow(XclosestObstacle-xKinectPerson,2)+pow(YclosestObstacle-
yKinectPerson,2))>radiusPerson) && (xKinectPerson!=0.0)) )){ //the closest obstacle inside the
DistanceCheck and more than radiusPerson from a detected legs or detected person from the Kinect
        linearCommand=0.2; //linear velocity near an obstacle
        if (YclosestObstacle>=0){angularCommand=-(WidthCheck-YclosestObstacle)/2;} //if obstacle
from the left than turn right (positive angular velocity)
        else {angularCommand=(WidthCheck+YclosestObstacle)/2;} //if obstacle from the right than
turn left (negative angular velocity)
        else if(linearVelocity>0.3){
            linearCommand=0.3; //if the linear velocity above 0.3 than slowdown to 0.3
            angularCommand=angularVelocity; SlowDown=true;}
    }
}

```

```

        else {linearCommand=linearVelocity;} //if linear velocity below 0.3- keep same velocity
        ob_msg.detect_obstacles=obstacle; //publish the variables
        ob_msg.slow_down=SlowDown;
        ob_msg.angular_velocity= angularCommand;
        ob_msg.linear_velocity= linearCommand;
        pub.publish(ob_msg);}

void velocityCallback(const geometry_msgs::Twist::ConstPtr& msg)
{angularVelocity=msg->angular.z; linearVelocity=msg->linear.x;}//the robot's velocity
};

int main(int argc, char **argv)
{
    ros::init(argc, argv, "laser_obstacles_avoidance");
    LaserObstacles LO;
    ros::NodeHandle n;
    ros::spin();
    return 0;
}

```

## 5. Search After Disappear

Implemented in 7.Direct Following Method and 8.History Following Method

## 6. Kinect Orientation Control – Pan Mechanism

```

#include "ros/ros.h" #include <std_msgs/Float64.h> #include <std_msgs/Float32.h>
#include "std_msgs/String.h" #include "opt_msgs/TrackArray.h"
#include "sensor_msgs/JointState.h" #include <ros/console.h> #include "math.h"
#include "trajectory_msgs/JointTrajectory.h" #include "nav_msgs/Odometry.h"
#include "trajectory_msgs/JointTrajectoryPoint.h"

class PanMove
{
    ros::NodeHandle n;
    ros::Time lastTrackTime; //last time that a person was detected
    ros::Subscriber sub1;
    ros::Publisher pub;
    ros::Subscriber sub2;
    double ConfidenceThreshold=1.1; //the SVM+HOG classifier- confidence for a real person
    double HeightThreshold=1.4; //height in meter of the person (minimum)
    double HeightMaxThreshold=2.0; //height in meter of the person (maximum)
    int TrackedID=0; //init the track ID to zero
    std_msgs::Float32 error_command;
    double AngleErrorPan; //the angle of the Pan related to the center of robot
    bool validTrack=false; //true if there is a valid track
    double AngleError=0; //the angle of the person related to the center of the Kinect
    bool TrackInitialized=false; //init there is no track

public:
    PanMove()
    {sub1 = n.subscribe("/tracker/tracks", 10, &PanMove::personCallback, this); //Open_PTrack
    pub = n.advertise<std_msgs::Float32>("/Pan_Error_Command", 1); //AngleError
    sub2 = n.subscribe("/Pan_Feedback", 10, &PanMove::panCallback, this);} //AngleErrorPan

    void panCallback(const std_msgs::Float32::ConstPtr& msg)
    {AngleErrorPan=msg->data;} //get the angle of the Pan related to the center of robot

    void personCallback(const opt_msgs::TrackArray::ConstPtr& msg)
    {validTrack=false;
    AngleError=0;
    int nbOfTracks=msg->tracks.size(); //Get the number of tracks in the TrackArray
    if (nbOfTracks>0) { //If at least 1 track, proceed
        if (!TrackInitialized){
            for(int i=0;i<nbOfTracks;i++){
                if ((msg->tracks[i].confidence>ConfidenceThreshold) && (msg->tracks[i].height>HeightThreshold) && (msg->tracks[i].height<HeightMaxThreshold)){ //oldest track
                    which is older than the age threshold, above the confidence threshold, above the height
                    threshold and under max height threshold
                    TrackedID=msg->tracks[i].id;
                    TrackInitialized=true;}
            }
        }
        if (!TrackInitialized){ROS_INFO("No valid track found");}
    }
}

```

```

        else
        {
            for(int i=0;i<nbOfTracks && !validTrack;i++){
                if (msg->tracks[i].id==TrackedID){
                    AngleError=atan2(msg->tracks[i].y,msg->tracks[i].x);//calculate
                    ROS_INFO("Error: %f", AngleError); validTrack=true; //stop for loop
                    lastTrackTime=ros::Time::now();}
            }
        }
    }
    ROS_INFO("valid %d" , validTrack);
    error_command.data=AngleError; //get the angle error of the person to send for twist
    if (!validTrack){ //no valid track
        error_command.data=-0.5*AngleErrorPan; //get opposite half of the angle of the pan (the
        pan will return slowly to the center of the robot)
        if ((ros::Time::now()-lastTrackTime)>ros::Duration(3)) //more than 3 seconds without
        a valid track
            {TrackInitialized=false;
            ROS_INFO("3 sec since last track seen, try to find it");}
    }
    pub.publish(error_command); //publish the command to the Pan to twist
}
};

int main(int argc, char **argv){
    ros::init(argc, argv, "orientation_control");
    PanMove pm;
    ros::NodeHandle n;
    ros::spin();
    return 0;
}

```

## 7. Direct Following Method

```

#include "std_msgs/String.h" #include "std_msgs/Float32.h" #include "nav_msgs/Odometry.h"
#include "geometry_msgs/Twist.h" #include "sensor_msgs/LaserScan.h"
#include "opt_msgs/TrackArray.h" #include <ros/console.h>
#include "people_msgs/PositionMeasurementArray.h" #include <obstacles/laserObstacles.h>
#include "people_msgs/PositionMeasurement.h" #include <occlusions/sideOcclusions.h>
#include "visualization_msgs/Marker.h" #include "visualization_msgs/MarkerArray.h"
#include <pcl_conversions/pcl_conversions.h> #include <pcl/point_types.h>
#include <pcl/PCLPointCloud2.h> #include <pcl/conversions.h> #define PI 3.14159265
geometry_msgs::Twist cmd_vel;

class kinect2_pan_laser
{
    ros::NodeHandle n; ros::Subscriber sub1; ros::Subscriber sub2; ros::Subscriber sub3;
    ros::Subscriber sub4; ros::Subscriber sub5; ros::Subscriber sub6; ros::Subscriber sub7;
    ros::Publisher cmd_vel_pub; ros::Publisher vis_pub1; ros::Publisher vis_pub2;
    ros::Publisher vis_pub3;
    double KpAngle=0.5; //the twist controller
    double KpAngleOcclusion=0.2; //for changing the following angle while occlusion
    double KpDistance=0.2; //the distance controller
    double DistanceTarget=1.2; //the minimum distance from the person
    double MaxSpeed=0.3; //the maximum linear speed
    double MaxTurn=0.2; //the maximum angular speed
    double AgeThreshold=0; //the "age" of the person (time that been detected)
    double ConfidenceThreshold=1.1; //SVM+HOG classifier- confidence for a real person
    double HeightThreshold=1.4; //height in meter of the person (minimum)
    double HeightMaxThreshold=2.0; //height in meter of the person (maximum)
    double AngleErrorPan=0; //the angle of the Pan related to the center of the robot
    bool smallError=false; //declare a small error to avoid small movements of the Pan
    double smallErrorThreshold=0.01; //threshold for avoid small movements of the Pan
    double AngleSmallError=0; //the angle of the person related to the Kinect center
    double xLaserPerson; double yLaserPerson; //position of the person from the laser
    double followingAngle=0; //15 deg= 0.2618 ,30 deg= 0.5236 rad, 60 deg= 1.0472 rad
    bool kinectLaserMatch=false; //the positions in both sensors correlate under 20 cm
    int nbOfTracksKinect=0; //number of people detection by the kinect
    double xRobot;double yRobot;double orientationRobot;//robot pose and orientation
    bool BigLeft; bool SmallLeft; bool WallLeft; //occlusions from the left side
    bool BigRight; bool SmallRight; bool WallRight; //occlusions from the right side
    bool laser_obstacle_flag; //true if an obstacle was found
    bool slow_down_flag; //true if an obstacle was found in slowdown distance
    double laser_angular_velocity=0; double laser_linear_velocity=0; //init the
    velocities from obstacles avoidance algorithm to zero
    double distanceKinect; //the distance of the person by the kinect
    double DistanceError; //error between the Kinect distance and distanceTarget
    double tempDistanceKinect; //temp distance to for Search After Disappear algorithm
}

```



```

double xPath; double yPath; //position of a person by Kinect related to the world
double xLast1; double yLast1; //last position of a person by Kinect before loss
double yLast2; //Y coordinates of a person by Kinect 4 frames before loss
double yDirection; //the direction of person's disappear (subtract Y coordinates)
double tempDistance; //last distance of a person by Kinect before loss
bool validTrackKinect=false; //true if there is a valid track by kinect
double angular_command=0; //angular velocity to avoid obstacle
bool validTrackLaser=false; //true if there is a valid track by laser
double tempDistanceLaser; //last distance of a person by laser before loss
double DistanceErrorLaser; //error between the laser distance and distanceTarget
int nbOfTracksLaser=0; //number of people detection by the laser
double AngleErrorFollow; //angle of last position of a person by kinect to robot
double AngleErrorLaser; //angle of position of a person by laser to robot
double xperson; double yperson; //position of a person by Kinect related to robot
double AngleErrorKinect; //angle of position of a person by kinect to robot
double age; //"age" of the person by Kinect (time that been detected)
double height; //height of the person by kinect
double confidence; //SVM+HOG classifier- confidence for a real person
double error; //error distance between the two sensor that detect a person
bool kinectTrack=false; //true if there is a valid track by kinect
bool laserTrack=false; //true if there is a valid track by laser
std::vector<double> YpathPoints; //vector of Y coordinates of a person by kinect

public:
    kinect2_pan_laser()
    {
        sub1= n.subscribe("/tracker/tracks", 10, &kinect2_pan_laser::personCallback, this);
//the kinect parameters of the person
        sub2= n.subscribe("/Pan_Feedback", 10, &kinect2_pan_laser::panCallback, this);
//the angle of the pan from the center of the robot
        sub3= n.subscribe("/Pan_Error_Command", 10, &kinect2_pan_laser::smallErrorCallback,
this); //the angle of a person from the center of the kinect
        sub4= n.subscribe("/people_tracker_measurements", 10,
&kinect2_pan_laser::LaserLegsCallback, this); //the laser parameters of a person
        sub5= n.subscribe("/occlusions/sideOcclusions", 10,
&kinect2_pan_laser::occlusionKinectCallback, this); //occlusions from depth Occlusion
        sub6= n.subscribe("/obstacles/laserObstacles", 10,
&kinect2_pan_laser::LaserObstaclesCallback, this); //obstacles from Obstacles Avoidance
        sub7= n.subscribe("/RosAria/pose", 10, &kinect2_pan_laser::poseCallback, this);
//position of the robot in the world
        cmd_vel_pub = ros::Publisher(n.advertise<geometry_msgs::Twist> ("follower/cmd_vel",
2)); //publish the velocities to the robot
        vis_pub1 = ros::Publisher(n.advertise<visualization_msgs::Marker>(
"/visualization_marker_array", 1 )); //for laser legs (green)
        vis_pub2 = ros::Publisher(n.advertise<visualization_msgs::Marker>(
"/visualization_marker_array", 1 )); //for Kinect person detected (blue)
        vis_pub3 = ros::Publisher(n.advertise<visualization_msgs::Marker>(
"/visualization_marker_array", 1 )); //for robot position (red)
    }

void poseCallback(const nav_msgs::Odometry::ConstPtr& msg)
{
xRobot=msg->pose.pose.position.x; yRobot=msg->pose.pose.position.y; //robot's position
tf::Pose pose; tf::poseMsgToTF(msg->pose.pose, pose);
orientationRobot= tf::getYaw(pose.getRotation()); //get radiand rotation (0 front, 3.14
back, left positive, right negative)
    ROS_INFO("xRobot: %f", xRobot); //print the values of the parameters
    ROS_INFO("yRobot: %f", yRobot); ROS_INFO("BigLeft: %d", BigLeft);
    ROS_INFO("SmallLeft: %d", SmallLeft); ROS_INFO("WallLeft: %d", WallLeft);
    ROS_INFO("BigRight: %d", BigRight); ROS_INFO("SmallRight: %d", SmallRight);
    ROS_INFO("WallRight: %d", WallRight); ROS_INFO("Height: %f", height);
    ROS_INFO("laser_obstacle_flag: %d", laser_obstacle_flag);
    ROS_INFO("xLaser: %f", xLaserPerson); ROS_INFO("yLaser: %f", yLaserPerson);
    ROS_INFO("match: %d", kinectLaserMatch); ROS_INFO("Confidence: %f", confidence);
    ROS_INFO("distanceKinect: %f", distanceKinect);
    ROS_INFO("AngleErrorPan: %f", (AngleErrorPan*180)/PI);
    ROS_INFO("xKinect: %f", xperson); ROS_INFO("yKinect: %f", yperson);
    ROS_INFO("xPath: 0"); ROS_INFO("yPath: 0"); ROS_INFO("tempDistance: %f", tempDistance);
    ROS_INFO("xFollow: 0"); ROS_INFO("yFollow: 0");
    ROS_INFO("kinectTrack: %d", kinectTrack); ROS_INFO("laserTrack: %d", laserTrack);
    for(int i=0;i<100000;i++){ //robot's position marker
        visualization_msgs::Marker marker; marker.header.frame_id = "odom";
        marker.header.stamp = ros::Time();marker.ns = "robotPose"; marker.id = i;
        marker.type = visualization_msgs::Marker::SPHERE;
        marker.action = visualization_msgs::Marker::ADD;
        marker.lifetime=ros::Duration(100.0);
        marker.pose.position.x = xRobot; marker.pose.position.y = yRobot;
        marker.pose.position.z = 0; marker.pose.orientation.x = 0.0;
        marker.pose.orientation.y = 0.0; marker.pose.orientation.z = 0.0;
        marker.pose.orientation.w = 1.0; marker.scale.x = 0.2; marker.scale.y = 0.2;
    }
}

```

```

    marker.scale.z = 0.2; marker.color.a = 1.0; marker.color.r = 1.0;
    marker.color.g = 0.0; marker.color.b = 0.0; vis_pub3.publish( marker );}
}

void occlusionKinectCallback(const occlusions::sideOcclusions::ConstPtr& msg)
{BigLeft= msg->bigLeft; //get all the Depth Occlusion algorithm parameters
SmallLeft= msg->smallLeft; WallLeft= msg->wallLeft; BigRight= msg->bigRight;
SmallRight= msg->smallRight; WallRight= msg->wallRight;
if (BigLeft && !BigRight && !SmallRight){followingAngle=-0.5236;} //change the following
angle according to the occlusion
if (SmallLeft && !BigLeft && !BigRight && !SmallRight){followingAngle=-0.2618;}
if (WallLeft && !WallRight){followingAngle=-0.2618;}
if (BigRight && !BigLeft && !SmallLeft){followingAngle=0.5236;}
if (SmallRight && !BigRight && !BigLeft && !SmallLeft){followingAngle=0.2618;}
if (WallRight && !WallLeft){followingAngle=0.2618;}
}

void LaserObstaclesCallback(const obstacles::laserObstacles::ConstPtr& msg)
{laser_obstacle_flag=msg->detect_obstacles;//get Obstacle Avoidance algorithm parameters
laser_angular_velocity=msg->angular_velocity;laser_linear_velocity=msg->linear_velocity;
slow_down_flag=msg->slow_down;
if (laser_obstacle_flag){
    cmd_vel.angular.z = laser_angular_velocity; //turn to avoid obstacles
    cmd_vel.linear.x = laser_linear_velocity; cmd_vel_pub.publish(cmd_vel);}
}

void smallErrorCallback(const std_msgs::Float32::ConstPtr& msg)
{AngleSmallError=msg->data; //get the angle of the person related to the kinect
if ((abs(AngleSmallError)<smallErrorThreshold)&&
(abs(AngleErrorPan)<0.01)){smallError=true;}; //avoid small movements of the Pan
    else {smallError=false;}
}

void LaserLegsCallback(const people_msgs::PositionMeasurementArray::ConstPtr& msg)
{validTrackLaser=false;
nbOfTracksLaser=msg->people.size();//number of people detection by the laser
if (nbOfTracksLaser>0) {
    xLaserPerson=msg->people[0].pos.x; //position of first detected person by laser
    yLaserPerson=msg->people[0].pos.y;
    if (nbOfTracksKinect==0) { //if there is no Kinect detection
        AngleErrorLaser=atan2(yLaserPerson,xLaserPerson); //Calculate angle error by laser
        DistanceErrorLaser=sqrt(pow(xLaserPerson,2)+pow(yLaserPerson,2)); //distance error
        if(!laser_obstacle_flag){ //no obstacle
            angular_command=AngleErrorLaser*KpAngle; //angular velocity depends on the twist
            controller and the angle error by laser
            if(angular_command>MaxTurn){angular_command=MaxTurn;} //limit maximum speed
            if(angular_command<-MaxTurn){angular_command=-MaxTurn;}
            cmd_vel.angular.z = angular_command;
            double linearspeedLaser=(DistanceErrorLaser-DistanceTarget)*KpDistance; //linear
            velocity depends on the distance error, the distanceTarget and the distance controller
            if (linearspeedLaser>MaxSpeed) {linearspeedLaser=MaxSpeed;}//limit maximum speed
            if (linearspeedLaser<0){linearspeedLaser=0;}
            cmd_vel.linear.x = linearspeedLaser; cmd_vel_pub.publish(cmd_vel);}
        }
        validTrackLaser=true; laserTrack=true;
        visualization_msgs::Marker marker; //person's position by laser marker
        marker.header.frame_id = "base_link"; marker.header.stamp = ros::Time();
        marker.ns = "laser"; marker.id = 0;
        marker.action = visualization_msgs::Marker::ADD;
        marker.pose.position.x = xLaserPerson; marker.pose.position.y = yLaserPerson;
        marker.pose.position.z = 0; marker.pose.orientation.x = 0.0;
        marker.pose.orientation.y = 0.0; marker.pose.orientation.z = 0.0;
        marker.pose.orientation.w = AngleErrorLaser;
        marker.scale.x = 0.1; marker.scale.y = 0.1; marker.scale.z = 0.1;
        marker.color.a = 1.0; marker.color.r = 0.0; marker.color.g = 1.0;
        marker.color.b = 0.0; vis_pub1.publish( marker );}
    else if(!validTrackKinect){laserTrack=false;}
}

void panCallback(const std_msgs::Float32::ConstPtr& msg)
{ AngleErrorPan=msg->data;} //get the angle of the Pan related to the robot

void personCallback(const opt_msgs::TrackArray::ConstPtr& msg)
{validTrackKinect=false;
nbOfTracksKinect=msg->tracks.size(); //get the number of people by kinect
if (nbOfTracksKinect>0) { //if at least 1 track, proceed
    for(int i=0;i<nbOfTracksKinect && !validTrackKinect;i++){
        if ((msg->tracks[i].age>AgeThreshold) && (msg-
>tracks[i].confidence>ConfidenceTheshold) && (msg->tracks[i].height>HeightTheshold) && (msg->

```

```

>tracks[i].height<HeightMaxTheshold)){ //oldest track which older than age threshold, above the
confidence threshold, above the height threshold and under max height threshold
    distanceKinect=msg->tracks[i].distance; //person's distance from the kinect
    xperson=((msg->tracks[i].distance)*cos(AngleSmallError+AngleErrorPan));
    yperson=((msg->tracks[i].distance)*sin(AngleSmallError+AngleErrorPan));
//position of the person related to the robot
    AngleErrorKinect=atan2(yperson,xperson); //angle of a person to robot
    age=msg->tracks[i].age; //"age" of a person (time that been detected)
    height=msg->tracks[i].height; //height of the person by kinect
    confidence=msg->tracks[i].confidence; //confidence for a real person
    YpathPoints.insert(YpathPoints.begin(),yperson); //insert the Y coordinate
    if (YpathPoints.size(>)5){
        yLast1=YpathPoints.at(4); yLast2=YpathPoints.at(1); xLast1=xperson;
        tempDistanceKinect=distanceKinect; //get the last person's distance
        yDirection=yLast2-yLast1; //substract to get the direction
        YpathPoints.pop_back();} //clear space for more Y coordinate
    error= sqrt(pow(xperson-xLaserPerson,2)+pow(yperson-yLaserPerson,2));
//calculate the x and y error between the kinect and the laser
    if (error<0.2){kinectLaserMatch=true;} //true if under 20 cm deviation
    else{kinectLaserMatch=false;}
    DistanceError=distanceKinect-DistanceTarget; //Calculate distance error
    if (!laser_obstacle_flag){ //no obstacle
        angular_command= AngleErrorKinect*KpAngle; //angular velocity depends on the
twist controller and the angle error by kinect
        if(abs(followingAngle)>0.1 && abs(AngleErrorKinect)<0.2) {angular_command =
(AngleErrorKinect+followingAngle)*KpAngleOcclusion;} //angular velocity also depends on the
following angle if it bigger than absolute value of 0.1
        if(angular_command>MaxTurn){angular_command=MaxTurn;}//limit maximum speed
        if(angular_command<-MaxTurn){angular_command=-MaxTurn;}
        cmd_vel.angular.z = angular_command;
        if (DistanceError>0.05){//threshold for small distance error of 0.05 meter
            double command_speed=DistanceError*KpDistance; //linear velocity depends on
the distance error and the distance controller
            if (command_speed>MaxSpeed){command_speed=MaxSpeed;}//limit maximum speed
            if (command_speed<0){command_speed=0;} //Avoid going backward
            cmd_vel.linear.x = command_speed;}
        validTrackKinect=true; cmd_vel_pub.publish(cmd_vel);}
    }
}
kinectTrack=true;
visualization_msgs::Marker marker; //person's position by kinect marker
marker.header.frame_id = "base_link"; marker.header.stamp = ros::Time();
marker.ns = "kinect"; marker.id = 0;
marker.type = visualization_msgs::Marker::SPHERE;
marker.action = visualization_msgs::Marker::ADD;
marker.pose.position.x = xperson; marker.pose.position.y = yperson;
marker.pose.position.z = 0; marker.pose.orientation.x = 0.0;
marker.pose.orientation.y = 0.0; marker.pose.orientation.z = 0.0;
marker.pose.orientation.w = AngleErrorKinect; marker.scale.x = 0.1;
marker.scale.y = 0.1; marker.scale.z = 0.1; marker.color.a = 1.0;
marker.color.r = 0.0; marker.color.g = 1.0; marker.color.b = 1.0;
vis_pub2.publish( marker );}
else if(!validTrackLaser){kinectTrack=false;//no track in both sensors (kinect and laser)
xPath= xRobot+cos(orientationRobot+AngleErrorKinect)*tempDistanceKinect;
yPath= yRobot+sin(orientationRobot+AngleErrorKinect)*tempDistanceKinect; //last
person's position related to the world
    AngleErrorFollow=-atan2(yPath-yRobot,(xPath-xRobot))+orientationRobot; //angle
related to the world
    if(abs(AngleErrorFollow)>PI){ //normalize the angle between [-PI,+PI]
        if(AngleErrorFollow<0){AngleErrorFollow=AngleErrorFollow+2*PI;}
        else{AngleErrorFollow=AngleErrorFollow-2*PI;}
    }
    tempDistance=sqrt(pow(xPath-xRobot,2)+pow(yPath-yRobot,2)); //person's distance
from the last position to the robot
    if (!laser_obstacle_flag){ //no obstacle
        ros::Time start= ros::Time::now();
        while((ros::Time::now()-start<ros::Duration(round(tempDistance/0.3))) &&
(!validTrackKinect) && (!laser_obstacle_flag)){ //no Kinect track, while time is shorter than
the distance of the last position divided by constast linear velocity of 0.3
            if (!laser_obstacle_flag)
                {cmd_vel.linear.x = 0.3; cmd_vel.angular.z=0.0;} //move straight
            else{cmd_vel.angular.z = laser_angular_velocity; //avoid obstacle
                cmd_vel.linear.x = laser_linear_velocity;}
            }
            cmd_vel.linear.x = 0.0;//robot reach to the last position of the person
            if(!validTrackKinect){ //if no Kinect track
                if(yDirection>0){cmd_vel.angular.z=0.2;}//robot turns to last direction
                else{cmd_vel.angular.z=-0.2;}
            }
        }
    }
}

```



```

        cmd_vel_pub.publish(cmd_vel);}
    }
};

int main(int argc, char **argv){
    ros::init(argc, argv, "simple_follower_kinect2_pan_laser");
    kinect2_pan_laser kpl;
    ros::NodeHandle n;
    ros::spin();
    return 0;
}

```

## 8. History Following Method

```

#include <stdio.h>      #include <stdlib.h>      #include "ros/ros.h"      #include "math.h"
#include "std_msgs/String.h" #include "std_msgs/Float32.h" #include "nav_msgs/Odometry.h"
#include "geometry_msgs/Twist.h" #include "sensor_msgs/LaserScan.h"
#include "opt_msgs/TrackArray.h" #include <ros/console.h>
#include "people_msgs/PositionMeasurementArray.h" #include <occlusions/sideOcclusions.h>
#include "people_msgs/PositionMeasurement.h" #include <obstacles/laserObstacles.h>
#include "visualization_msgs/Marker.h" #include "visualization_msgs/MarkerArray.h"
#include <pcl_conversions/pcl_conversions.h> #include <pcl/point_types.h>
#include <pcl/PCLPointCloud2.h> #include <pcl/conversions.h>
#include <pcl_ros/transforms.h> #include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp> #define PI 3.14159265
geometry_msgs::Twist cmd_vel;

class kinect2_pan_laser
{
    ros::NodeHandle n; ros::Subscriber sub1; ros::Subscriber sub2; ros::Subscriber sub3;
    ros::Subscriber sub4; ros::Subscriber sub5; ros::Subscriber sub6; ros::Subscriber sub7;
    ros::Publisher vis_pub1; ros::Publisher vis_pub2; ros::Publisher vis_pub3;
    ros::Publisher cmd_vel_pub;

    double KpAngle=0.5; //the twist controller
    double KpAngleOcclusion=0.2; //for changing the following angle while occlusion
    double KpDistance=0.2; //the distance controller
    double DistanceTarget=1.2; //the minimum distance from the person
    double MaxSpeed=0.3; //the maximum linear speed
    double MaxTurn=0.2; //the maximum angular speed
    double AgeThreshold=0; //the "age" of the person (time that been detected)
    double ConfidenceTheshold=1.1; //SVM+HOG classifier- confidence for a real person
    double HeightTheshold=1.4; //height in meter of the person (minimum)
    double HeightMaxTheshold=2; //height in meter of the person (maximum)
    double AngleErrorPan=0; //the angle of the Pan related to the center of the robot
    bool smallError=false; //declare a small error to avoid small movements of the Pan
    double smallErrorThreshold=0.01; //threshold for avoid small movements of the Pan
    double AngleSmallError=0; //the angle of the person related to the Kinect center
    double xLaserPerson; double yLaserPerson; //position of the person from the laser
    double linearspeedLaser; //linear speed that depends on the distance by laser
    double DistanceErrorLaser; //the distance of the person by the laser
    double DistanceErrorKinect; //the distance of the person by the kinect
    double linearspeedKinect; //linear speed that depends on the distance by kinect
    double xRobot; double yRobot; double orientationRobot; //robot's pose and orientation
    double xperson; double yperson; //position of a person by Kinect related to robot
    double xPath; double yPath; //position of a person by Kinect related to the world
    double distanceKinect; //the distance of the person by the kinect
    double xFollow; double yFollow; //person's historical position by Kinect
    double AngleErrorFollow; //angle of an historical position of a person by kinect
    double DistanceErrorFollow; //distance of an historical position of a person
    double followingAngle=0; //15 deg= 0.2618 ,30 deg= 0.5236 rad, 60 deg= 1.0472 rad
    bool kinectLaserMatch=false; //the positions in both sensors correlate under 20 cm
    int nbOfTracksKinect=0; //number of people detection by the kinect
    bool BigLeft; bool SmallLeft; bool WallLeft; //occlusions from the left side
    bool BigRight; bool SmallRight; bool WallRight; //occlusions from the right side
    bool laser_obstacle_flag; //true if an obstacle was found
    bool slow_down_flag; //true if an obstacle was found in slowdown distance
    double laser_angular_velocity=0; double laser_linear_velocity=0; //init the
    velocities from obstacles avoidance algorithm to zero
    double AngleErrorLaser; //angle of position of a person by laser to robot
    double AngleErrorKinect; //angle of position of a person by kinect to robot
    double error; //error distance between the two sensor that detect a person
    double age; // "age" of the person by Kinect (time that been detected)
    double height; //height of the person by kinect
    double confidence; //SVM+HOG classifier- confidence for a real person
    bool kinectTrack=false; //true if there is a valid track by kinect
    bool laserTrack=false; //true if there is a valid track by laser
}

```

```

std::vector<double> XpathPoints; //vector of X coordinates of a person
std::vector<double> YpathPoints; //vector of Y coordinates of a person
std::vector<double> AngleErrorPanHistory; //vector of angle of the Pan to the robot

public:
    kinect2_pan_laser()
    {
        sub1= n.subscribe("/tracker/tracks", 10, &kinect2_pan_laser::personCallback, this);
        //the kinect parameters of the person
        sub2= n.subscribe("/Pan_Feedback", 10, &kinect2_pan_laser::panCallback, this);
        //the angle of the pan from the center of the robot
        sub3= n.subscribe("/Pan_Error_Command", 10, &kinect2_pan_laser::smallErrorCallback,
this); //angle of a person from the kinect's center
        sub4= n.subscribe("/people_tracker_measurements", 10,
&kinect2_pan_laser::LaserLegsCallback, this); //the laser parameters of a person
        sub5= n.subscribe("/occlusions/sideOcclusions", 10,
&kinect2_pan_laser::occlusionKinectCallback, this); //occlusions from depth occlusion
        sub6= n.subscribe("/obstacles/laserObstacles", 10,
&kinect2_pan_laser::LaserObstaclesCallback, this); //obstacles from obstacle avoidance
        sub7= n.subscribe("/RosAria/pose", 10, &kinect2_pan_laser::poseCallback, this);
        //position of the robot in the world
        vis_pub1 = ros::Publisher(n.advertise<visualization_msgs::Marker>(
"/visualization_marker_array", 1 )); //for laser legs (green)
        vis_pub2 = ros::Publisher(n.advertise<visualization_msgs::Marker>(
"/visualization_marker_array", 1 )); //for Kinect person detected (blue)
        vis_pub3 = ros::Publisher(n.advertise<visualization_msgs::Marker>(
"/visualization_marker_array", 1 )); //for robot position (red), when using rviz set the fixed
frame to odom
        cmd_vel_pub = ros::Publisher(n.advertise<geometry_msgs::Twist> ("follower/cmd_vel",
2));
    }

void poseCallback(const nav_msgs::Odometry::ConstPtr& msg)
{
    xRobot=msg->pose.position.x; yRobot=msg->pose.position.y; //robot's position
    tf::Pose pose; tf::poseMsgToTF(msg->pose.pose, pose);
    orientationRobot= tf::getYaw(pose.getRotation()); //get radian rotation (0 front, 3.14 back,
left positive, right negative)
    ROS_INFO("xRobot: %f", xRobot); //print the values of the parameters
    ROS_INFO("yRobot: %f", yRobot); ROS_INFO("BigLeft: %d", BigLeft);
    ROS_INFO("SmallLeft: %d", SmallLeft); ROS_INFO("WallLeft: %d", WallLeft);
    ROS_INFO("BigRight: %d", BigRight); ROS_INFO("SmallRight: %d", SmallRight);
    ROS_INFO("WallRight: %d", WallRight);
    ROS_INFO("laser_obstacle_flag: %d", laser_obstacle_flag);
    ROS_INFO("xLaser: %f", xLaserPerson); ROS_INFO("yLaser: %f", yLaserPerson);
    ROS_INFO("match: %d", kinectLaserMatch); ROS_INFO("Confidence: %f", confidence);
    ROS_INFO("Height: %f", height); ROS_INFO("distanceKinect: %f", distanceKinect);
    ROS_INFO("AngleErrorPan: %f", (AngleErrorPan*180)/ PI); ROS_INFO("xKinect: %f", xperson);
    ROS_INFO("yKinect: %f", yperson); ROS_INFO("xPath: %f", xPath);
    ROS_INFO("yPath: %f", yPath); ROS_INFO("xFollow: %f", xFollow);
    ROS_INFO("yFollow: %f", yFollow); ROS_INFO("tempDistance: 0");
    ROS_INFO("kinectTrack: %d", kinectTrack); ROS_INFO("laserTrack: %d", laserTrack);
    for(int i=0; i<100000; i++){ //robot's position marker
        visualization_msgs::Marker marker; marker.header.frame_id = "odom";
        marker.header.stamp = ros::Time(); marker.ns = "robotPose";
        marker.id = i; marker.type = visualization_msgs::Marker::SPHERE;
        marker.action = visualization_msgs::Marker::ADD; marker.pose.position.x = xRobot;
        marker.pose.position.y = yRobot; marker.pose.position.z = 0;
        marker.pose.orientation.x = 0.0; marker.pose.orientation.y = 0.0;
        marker.pose.orientation.z = 0.0; marker.pose.orientation.w = 1.0;
        marker.scale.x = 0.2; marker.scale.y = 0.2; marker.scale.z = 0.2;
        marker.color.a = 1.0; marker.color.r = 1.0; marker.color.g = 0.0;
        marker.color.b = 0.0; vis_pub3.publish( marker );}
    }

void occlusionKinectCallback(const occlusions::sideOcclusions::ConstPtr& msg)
{
    BigLeft= msg->bigLeft; //get all the Depth Occlusion algorithm parameters
    SmallLeft= msg->smallLeft; WallLeft= msg->wallLeft; BigRight= msg->bigRight;
    SmallRight= msg->smallRight; WallRight= msg->wallRight;
    if (BigLeft && !BigRight && !SmallRight){followingAngle=-0.5236;} //change the following
angle according to the occlusion
    if (SmallLeft && !BigLeft && !BigRight && !SmallRight){followingAngle=-0.2618;}
    if (WallLeft && !WallRight){followingAngle=-0.2618;}
    if (BigRight && !BigLeft && !SmallLeft){followingAngle=0.5236;}
    if (SmallRight && !BigRight && !BigLeft && !SmallLeft){followingAngle=0.2618;}
    if (WallRight && !WallLeft){followingAngle=0.2618;}
}

void LaserObstaclesCallback(const obstacles::laserObstacles::ConstPtr& msg)
{
    laser_obstacle_flag=msg->detect_obstacles; //get Obstacle Avoidance algorithm parameters
}

```

```

laser_angular_velocity=msg->angular_velocity;laser_linear_velocity=msg->linear_velocity;
slow_down_flag=msg->slow_down;
if (laser_obstacle_flag){
cmd_vel.angular.z = laser_angular_velocity; //turn to avoid obstacles
cmd_vel.linear.x = laser_linear_velocity; cmd_vel_pub.publish(cmd_vel);}
}

void smallErrorCallback(const std_msgs::Float32::ConstPtr& msg)
{AngleSmallError=msg->data; //get the angle of the person related to the kinect
if ((abs(AngleSmallError)<smallErrorThreshold)&&
(abs(AngleErrorPan)<0.01)){smallError=true;} //avoid small movements of the Pan
else {smallError=false;}
}

void LaserLegsCallback(const people_msgs::PositionMeasurementArray::ConstPtr& msg)
{bool validTrackLaser=false;
int nbOfTracksLaser=msg->people.size();();//number of people detection by the laser
if (nbOfTracksLaser>0) {
xLaserPerson=msg->people[0].pos.x; //position of first detected person by laser
yLaserPerson=msg->people[0].pos.y;
if (nbOfTracksKinect==0) { //if there is no Kinect detecti
AngleErrorLaser=atan2(yLaserPerson,xLaserPerson); //Calculate angle error by laser
DistanceErrorLaser=sqrt(pow(xLaserPerson,2)+pow(yLaserPerson,2)); //distance error
xPath= xRobot+cos(orientationRobot+AngleErrorLaser)*DistanceErrorLaser; //person's
position related to the world
yPath= yRobot+sin(orientationRobot+AngleErrorLaser)*DistanceErrorLaser;
XpathPoints.insert(XpathPoints.begin(),xPath); //insert the X coordinate
YpathPoints.insert(YpathPoints.begin(),yPath); //insert the Y coordinate
if (XpathPoints.size()>31){ //30 equal to 4 second history (8 Hz)
xFollow=XpathPoints.at(30); //get the 4 seconds historical position (8 Hz)
yFollow=YpathPoints.at(30);
XpathPoints.pop_back(); //clear space for more coordinates
YpathPoints.pop_back();
if(!laser_obstacle_flag){ //no obstacle
AngleErrorFollow=atan2(yFollow-yRobot,(xFollow-xRobot))+orientationRobot; //angle
related to the world
if(abs(AngleErrorFollow)>PI){ //normalize the angle between [-PI,+PI]
if(AngleErrorFollow<0){AngleErrorFollow=AngleErrorFollow+2*PI;}
else{AngleErrorFollow=AngleErrorFollow-2*PI;}
}
double angular_command;
angular_command=-AngleErrorFollow*KpAngle; //angular velocity depends on the twist
controller and the angle error by laser
if(angular_command>MaxTurn){angular_command=MaxTurn;} //limit maximum speed
if(angular_command<-MaxTurn){angular_command=-MaxTurn;}
cmd_vel.angular.z = angular_command;
DistanceErrorFollow=sqrt(pow(xFollow-xRobot,2)+pow(yFollow-yRobot,2)); //distance related
to the world
if (DistanceErrorLaser>DistanceTarget){ linearspeedLaser=(DistanceErrorFollow-
DistanceTarget)*KpDistance;} //as long as the distance is bigger than distanceTarget, linear
velocity depends on the distance error and the distance controller
else{linearspeedLaser=0;}
if (linearspeedLaser>MaxSpeed) {linearspeedLaser=MaxSpeed;} //limit maximum speed
if (linearspeedLaser<0) {linearspeedLaser=0;} //avoid going backward
cmd_vel.linear.x = linearspeedLaser;}
}
cmd_vel_pub.publish(cmd_vel);
}
validTrackLaser=true; laserTrack=true;
visualization_msgs::Marker marker; //person's position by laser marker
marker.header.frame_id = "base_link";marker.header.stamp = ros::Time();
marker.ns = "laser"; marker.id = 0; marker.pose.position.x = xLaserPerson;
marker.type = visualization_msgs::Marker::SPHERE;
marker.action = visualization_msgs::Marker::ADD;
marker.pose.position.y = yLaserPerson; marker.pose.position.z = 0;
marker.pose.orientation.x = 0.0; marker.pose.orientation.y = 0.0;
marker.pose.orientation.z = 0.0;
marker.pose.orientation.w = AngleErrorLaser; marker.scale.x = 0.1;
marker.scale.y = 0.1; marker.scale.z = 0.1; marker.color.a = 1.0;
marker.color.r = 0.0; marker.color.g = 1.0; marker.color.b = 0.0;
vis_pub1.publish( marker );}
else{laserTrack=false;}
}

void panCallback(const std_msgs::Float32::ConstPtr& msg)
{AngleErrorPan=msg->data; //get the angle of the Pan related to the robot
AngleErrorPanHistory.insert(AngleErrorPanHistory.begin(),AngleErrorPan); //insert angle of
the Pan to the robot
if(AngleErrorPanHistory.size()>3){

```

```

        if(abs(AngleErrorPan)>2.0){AngleErrorPan=AngleErrorPanHistory.at(2);} //avoid angles
        above absolute value of 2 radians
        AngleErrorPanHistory.pop_back();}
    }

    void personCallback(const opt_msgs::TrackArray::ConstPtr& msg)
    {bool validTrack=false;
    nbOfTracksKinect=msg->tracks.size();//get the number of people by kinect
    if (nbOfTracksKinect>0) { //if at least 1 track, proceed
        for(int i=0;i<nbOfTracksKinect && !validTrack;i++){
            if ((msg->tracks[i].age>AgeThreshold) && (msg->tracks[i].confidence>ConfidenceTheshold) && (msg->tracks[i].height>HeightTheshold) && (msg->tracks[i].height<HeightMaxTheshold)){ //oldest track which older than age threshold, above the
            confidence threshold, above the height threshold and under max height threshold
                distanceKinect=msg->tracks[i].distance; //person's distance from the kinect
                xperson=((distanceKinect)*cos(AngleSmallError+AngleErrorPan));
                yperson=((distanceKinect)*sin(AngleSmallError+AngleErrorPan)); //position of the
                person related to the robot
                AngleErrorKinect=atan2(yperson,xperson); //angle of a person to robot
                age=msg->tracks[i].age; //"age" of a person (time that been detected)
                height=msg->tracks[i].height; //height of the person by kinect
                confidence=msg->tracks[i].confidence; //confidence for a real person
                xPath= xRobot+cos(orientationRobot+AngleErrorKinect)*distanceKinect;
                yPath= yRobot+sin(orientationRobot+AngleErrorKinect)*distanceKinect; //person's
                position related to the world
                XpathPoints.insert(XpathPoints.begin(),xPath); //insert the X coordinate
                YpathPoints.insert(YpathPoints.begin(),yPath); //insert the Y coordinate
                if (XpathPoints.size()>81){ //80 equal to 4 second history (20 Hz)
                    if(sqrt(pow(XpathPoints.at(80)-XpathPoints.at(79),2)+pow(YpathPoints.at(80)-
                    YpathPoints.at(79),2))<1.0){ //avoid big false position change, the position of a person can not
                    be change more than 1 meter at 1 frame
                        xFollow=XpathPoints.at(80); //get the 4 seconds historical position (20 Hz)
                        yFollow=YpathPoints.at(80);
                    }
                    XpathPoints.pop_back(); //clear space for more coordinates
                    YpathPoints.pop_back();
                    error= sqrt(pow(xperson-xLaserPerson,2)+pow(yperson-yLaserPerson,2));
                    //calculate the x and y error between the kinect and the laser
                    if (error<0.2) {kinectLaserMatch=true;} //true if under 20 cm deviation
                    else{kinectLaserMatch=false;}
                    DistanceErrorKinect=msg->tracks[i].distance; //person's distance from kinect
                    if (!laser_obstacle_flag){ //no obstacle
                        AngleErrorFollow=-atan2(yFollow-yRobot,(xFollow-xRobot))+orientationRobot;
                        //angle related to the world
                        if(abs(AngleErrorFollow)>PI){ //normalize the angle between [-PI,+PI]
                            if(AngleErrorFollow<0){AngleErrorFollow=AngleErrorFollow+2*PI;}
                            else{AngleErrorFollow=AngleErrorFollow-2*PI;}
                        }
                        double angular_command;
                        if(abs(followingAngle)<0.1){angular_command = (-
                        AngleErrorFollow+followingAngle)*KpAngle;} //angular velocity also depends on the following
                        angle if it bigger than absolute value of 0.1
                        else {angular_command =-AngleErrorFollow*KpAngleOcclusion;} //angular
                        velocity depends on the twist occlusion controller and the angle error by kinect
                        if(angular_command>MaxTurn){angular_command=MaxTurn;} //limit speed
                        if(angular_command<-MaxTurn){angular_command=-MaxTurn;}
                        cmd_vel.angular.z = angular_command;
                        DistanceErrorFollow=sqrt(pow(xFollow-xRobot,2)+pow(yFollow-yRobot,2));
                        //person's distance from the robot
                        if (DistanceErrorKinect>DistanceTarget){ //as long as distance is bigger than
                        distanceTarget, linear velocity depends on distance error and distance controller
                        linearspeedKinect=(DistanceErrorFollow-DistanceTarget)*KpDistance;}
                        else{linearspeedKinect=0;}
                        if (linearspeedKinect>MaxSpeed) {linearspeedKinect=MaxSpeed;} //limit speed
                        if (linearspeedKinect<0 || DistanceErrorKinect<0.05 ) //avoid going backward and
                        when reach to 5 cm threshold of disyanceTarget
                        {linearspeedKinect=0;}
                        cmd_vel.linear.x = linearspeedKinect;
                        validTrack=true; cmd_vel_pub.publish(cmd_vel);}
                    }
                }
                kinectTrack=true;
                visualization_msgs::Marker marker; //person's position by kinect marker
                marker.header.frame_id = "base_link";
                marker.header.stamp = ros::Time(); marker.ns = "kinect"; marker.id = 0;
                marker.type = visualization_msgs::Marker::SPHERE;
                marker.action = visualization_msgs::Marker::ADD;
                marker.pose.position.x = xperson; marker.pose.position.y = yperson;
                marker.pose.position.z = 0; marker.pose.orientation.x = 0.0;

```

```

        marker.pose.orientation.y = 0.0;    marker.pose.orientation.z = 0.0;
        marker.pose.orientation.w = AngleErrorKinect;    marker.scale.x = 0.1;
        marker.scale.y = 0.1; marker.scale.z = 0.1;    marker.color.a = 1.0;
        marker.color.r = 0.0; marker.color.g = 1.0;    marker.color.b = 1.0;
        vis_pub2.publish( marker );}
    }
    else { kinectTrack=false;
    if (!laser_obstacle_flag && !laserTrack){ //no obstacle and no Kinect track
        ros::Time start= ros::Time::now();
        while((ros::Time::now()-start<ros::Duration(round(tempDistance/0.3))) &&
(!kinectTrack) && (!laser_obstacle_flag)){ //no Kinect track, while time is shorter than the
distance of the last position divided by constant linear velocity of 0.3
            if (!laser_obstacle_flag)
                {cmd_vel.linear.x = 0.3; cmd_vel.angular.z=0.0;} //move straight
            else{cmd_vel.angular.z = laser_angular_velocity; //avoid obstacle
                cmd_vel.linear.x = laser_linear_velocity;}
            }
            cmd_vel.linear.x = 0.0; //robot reach to the last position of the person
            if(!kinectTrack){ //if no Kinect track
                if(yDirection>0){cmd_vel.angular.z=0.2;}//robot turns to last direction
                else{cmd_vel.angular.z=-0.2;}
            }
            cmd_vel_pub.publish(cmd_vel);}
        }
    }
};

int main(int argc, char **argv){
    ros::init(argc, argv, "simple_follower_kinect2_pan_laser");
    kinect2_pan_laser kpl;
    ros::NodeHandle n;
    ros::spin();
    return 0;
}

```

## 9. Adaptive Following Method (Kinect and Laser)

Implemented in 7.Direct Following Method and 8.History Following Method

שני הניסויים הראשונים היו ניסויים ראשוניים כדי לבחור את פרמטרי העקיבה של הרובוט. לאחר שילוב קינקט V2 (הגרסה החדשה של הקינקט) עם מנגנון סיבוב הקינקט (Pan), נערך ניסוי לבחינת תוצאות אובייקטיביות וסובייקטיביות של עקיבה בשלוש זוויות שונות ( $0^\circ$ ,  $30^\circ$ ,  $60^\circ$ ). התוצאות הצביעו על כך שאין הבדל משמעותי בין עקיבה בזווית  $0^\circ$  (ישירות מאחורי האדם) לבין עקיבה בזווית  $30^\circ$ . התוצאות עבור עקיבה בזווית  $60^\circ$  לא היו אמינות מספיק.

בניסוי שמטרתו להשוות בין אלגוריתמי זיהוי ההסתרות השונים, **זיהוי הסתרות עומק** הניב את הביצועים הטובים ביותר. בניסוי כדי לקבוע את השילוב הטוב ביותר של שלוש האלגוריתמים (**הימנעות ממכשולים**, **זיהוי הסתרות עומק** ו**חיפוש אחרי העלמות**), פעם אחת עם שיטת **העקיבה הישירה** ופעם עם **העקיבה ההיסטורית**, עולה כי השילוב של שלושת האלגוריתמים נתן את הביצועים הטובים ביותר. הניסוי האחרון והחשוב ביותר, השווה בין שתי שיטות העקיבה (**עקיבה ישירה** ו**עקיבה היסטורית**) עם שילוב של שלושת האלגוריתמים (**הימנעות ממכשולים**, **זיהוי הסתרות עומק** ו**חיפוש אחרי העלמות**) לאותן שיטות עקיבה עם אותם אלגוריתמים בתוספת לייזר לגילוי רגליים (נקראת שיטת מסתגלת אדפטיבית) לשימוש במידת הצורך אם הקינקט מאבד את האדם.

**מסקנות.** התוצאות הראו כי השיטות המסתגלות (אדפטיביות) המשלבות את חיישן הקינקט וחיישן הלייזר לעקיבה אחרי האדם היו יותר טובות מהשיטות הלא מסתגלות (רק עם חיישן הקינקט לעקיבה אחר האדם) ו**עקיבה ישירה** עדיפה על **עקיבה היסטורית**.



## תקציר

יכולתו של הרובוט לעקוב אחר אדם בסביבה פנימית לא ממופה טומן בחובו אתגרים בשל מכשולים לא ידועים, קירות לא חקורים, פינות לא מוכרות ומסדרונות. מטרת המחקר הנוכחי היא לפתח אלגוריתמי עקיבה אחרי אדם באמצעות רובוט אשר מפחיתים את מספר מקרי אובדן העקיבה אחר האדם כדי לשפר את יכולת הרובוט להתאוששות עצמאית בסביבה לא ידועה. לשם כך, חמישה אלגוריתמים ושתי שיטות עקיבה פותחו ונבדקו בסדרת ניסויים על פלטפורמת רובוט נייד.

אלגוריתמים. האלגוריתמים שפותחו לא משתמשים במידע מקדים על הסביבה (למשל, פועלים ללא מיפוי מקדים) ואינם דורשים מהאדם לשאת פריט ספציפי או בגדים ספציפיים.

האלגוריתם משתמש בשיטות המבוססות על ראיית עומק כדי לשפר את תהליך זיהוי ההסתרה. האלגוריתם משתמש בחיפוש לייזר כדי להימנע ממכשולים במהלך העקיבה בזמן אמת, בהתאם למהירות הלינארית והזוויתית של הרובוט וזוכר את המיקום האחרון של האדם כדי לחפש אותו אם נעלם על ידי תזוזת הרובוט למיקום האחרון של האדם ופנייה לכיוון שחושב לפני ההעלמות.

חמשת האלגוריתמים שפותחו: **הימנעות ממכשולים (Obstacles-Avoidance- OA)** בזמן אמת באמצעות לייזר, **חיפוש אחרי העלמות (Search-After-Disappear- SAD)** לאחר איבוד עקיבה והעלמות האדם, ושלושה אלגוריתמי זיהוי הסתרות, **זיהוי הסתרת עומק (-Depth-Occlusion- DO)** באמצעות מידע על מרחק הפיקסלים של הקינקט, **זיהוי הסתרת ראייה (-Vision- DO)** באמצעות מידע דו ממדי של הקינקט ו**זיהוי הסתרה משולב (Occlusion-Detection- VO)** באמצעות שני הסנסורים של הקינקט שתוארו.

חיישנים. שני חיישנים שימשו לזיהוי בני אדם: (1) קינקט באמצעות OpenPTrack (<http://openptrack.org/>) המזהה אנשים העומדים על הקרקע באמצעות היסטוגרמה של אוריינטציה הדרגתית (histogram of oriented gradients- HOG) ומכונת ווקטורים תומכים (support vector machine- SVM). (2) הלייזר המובנה על הרובוט SICK300, אשר יכול לזהות את הרגליים של האדם ב-20 סנטימטר מעל פני הקרקע.

שיטות עקיבה. שתי שיטות עקיבה עיקריות פותחו והוערכו. **עקיבה ישירה (Direct-Following- DF)**, אשר גורמת לרובוט לנוע ישירות לכיוון האדם **עקיבה היסטורית (History-Following- HF)** אשר גורמת לרובוט לנוע למיקומים היסטוריים (קודמים) של האדם. ההערכה של שיטות העקיבה בוצעה בשני אופנים: (1) עקיבה באמצעות זיהוי האדם רק על ידי הקינקט, נקראת שיטה לא מסתגלת (לא אדפטיבית). (2) עקיבה באמצעות זיהוי האדם על ידי שני החיישנים (קינקט ולייזר). כאשר הקינקט מאבד עקיבה אז שיטת העקיבה עוברת לחיפוש הלייזר לזיהוי האדם, נקראת שיטה מסתגלת (אדפטיבית).

פלטפורמת הרובוט. האלגוריתמים יושמו על פלטפורמת רובוט ניידת Pioneer LXRobot מצוידת בקינקט וחיישן לייזר.

ניסויים- שיטות ותוצאות. כדי להתגבר על הקשיים הכרוכים בסביבה לא מוכרת, אלגוריתמים הפועלים בזמן אמת פותחו ושולבו בצירופים שונים עם שתי שיטות העקיבה הראשיות (**עקיבה ישירה** ו**עקיבה היסטורית**). סדרת ניסויים נערכה לטובת בחירת הפרמטרים הטובים ביותר והערכה של ביצועי האלגוריתמים. מדדי הביצוע שנבחרו לצורך השוואת הביצועים היו מספר מקרי איבוד העקיבה אחר האדם, מספר הפעמים בהן העקיבה התחדשה בעצמה על ידי הרובוט ללא התערבות המפעיל, מספר התערבויות המפעיל לטובת חידוש עקיבה ומספר התערבויות המפעיל לצורכי בטיחות, המרחק בין הרובוט לבין האדם, אורך נתיב הרובוט, האמינות של זיהוי הרגליים, האמינות של גילוי ההסתרות, והיחס של עקיבה יציבה לעומת אי עקיבה (אחוז הזמן בו החיישן עקב באופן יציב אחר האדם מסך זמן הניסוי כולו) של הקינקט ושל הלייזר.



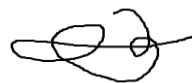


אוניברסיטת בן גוריון בנגב  
הפקולטה למדעי ההנדסה  
המחלקה לתעשייה וניהול

## **פיתוח אלגוריתמים עבור רובוט מצוייד בחיישני קינקט ולייזר בסביבה פנימית לא ידועה עם מכשולים ופינות אשר עוקב אחר אדם**

חיבור זה מהווה חלק מהדרישות לקבלת התואר "מגיסטר" בהנדסה

דרור כץ



חתימת הסטודנט : דרור כץ



חתימת המנחה : פרופ' יעל אידן



חתימת יו"ר ועדה המחלקתית : ד"ר ישראל פרמט

יולי 2016

אוניברסיטת בן גוריון בנגב  
הפקולטה למדעי ההנדסה  
המחלקה לתעשייה וניהול

**פיתוח אלגוריתמים עבור רובוט מצוייד  
בחיישני קינקט ולייזר בסביבה פנימית לא  
ידועה עם מכשולים ופינות אשר עוקב אחר  
אדם**

חיבור זה מהווה חלק מהדרישות לקבלת התואר "מגיסטר" בהנדסה

דרור כץ  
מנחה : פרופ' יעל אידן

יולי 2016

באר-שבע