

5-2017

A Voice and Pointing Gesture Interaction System for Supporting Human Spontaneous Decisions in Autonomous Cars

Pablo Sauras-Perez

Clemson University, psauras@g.clemson.edu

Follow this and additional works at: https://tigerprints.clemson.edu/all_dissertations

Recommended Citation

Sauras-Perez, Pablo, "A Voice and Pointing Gesture Interaction System for Supporting Human Spontaneous Decisions in Autonomous Cars" (2017). *All Dissertations*. 1943.

https://tigerprints.clemson.edu/all_dissertations/1943

This Dissertation is brought to you for free and open access by the Dissertations at TigerPrints. It has been accepted for inclusion in All Dissertations by an authorized administrator of TigerPrints. For more information, please contact kokeefe@clemson.edu.

A VOICE AND POINTING GESTURE INTERACTION SYSTEM FOR SUPPORTING
HUMAN SPONTANEOUS DECISIONS IN AUTONOMOUS CARS

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Automotive Engineering

by
Pablo Sauras-Perez
May 2017

Accepted by:
Dr. Pierluigi Pisu, Committee Chair
Dr. Joachim Taiber
Dr. Yunyi Jia
Dr. Mashrur Chowdhury

ABSTRACT

Autonomous cars are expected to improve road safety, traffic and mobility. It is projected that in the next 20-30 years fully autonomous vehicles will be on the market.

The advancement on the research and development of this technology will allow the disengagement of humans from the driving task, which will be responsibility of the vehicle intelligence. In this scenario new vehicle interior designs are proposed, enabling more flexible human vehicle interactions inside them. In addition, as some important stakeholders propose, control elements such as the steering wheel and accelerator and brake pedals may not be needed any longer.

However, this user control disengagement is one of the main issues related with the user acceptance of this technology. Users do not seem to be comfortable with the idea of giving all the decision power to the vehicle. In addition, there can be location awareness situations where the user makes a spontaneous decision and requires some type of vehicle control. Such is the case of stopping at a particular point of interest or taking a detour in the pre-calculated autonomous route of the car.

Vehicle manufacturers' maintain the steering wheel as a control element, allowing the driver to take over the vehicle if needed or wanted. This causes a constraint in the previously mentioned human vehicle interaction flexibility.

Thus, there is an unsolved dilemma between providing users enough control over the autonomous vehicle and route so they can make spontaneous decision, and interaction flexibility inside the car.

This dissertation proposes the use of a voice and pointing gesture human vehicle interaction system to solve this dilemma. Voice and pointing gestures have been identified as natural interaction techniques to guide and command mobile robots, potentially providing the needed user control over the car. On the other hand, they can be executed anywhere inside the vehicle, enabling interaction flexibility.

The objective of this dissertation is to provide a strategy to support this system. For this, a method based on pointing rays intersections for the computation of the point of interest (POI) that the user is pointing to is developed. Simulation results show that this POI computation method outperforms the traditional ray-casting based by 76.5% in cluttered environments and 36.25% in combined cluttered and non-cluttered scenarios. The whole system is developed and demonstrated using a robotics simulator framework. The simulations show how voice and pointing commands performed by the user update the predefined autonomous path, based on the recognized command semantics. In addition, a dialog feedback strategy is proposed to solve conflicting situations such as ambiguity in the POI identification. This additional step is able to solve all the previously mentioned POI computation inaccuracies. In addition, it allows the user to confirm, correct or reject the performed commands in case the system misunderstands them.

DEDICATION

Papa, Mama, Borja, Carlos ... Family ... Friends

... This is for you.

Gracias!

ACKNOWLEDGMENTS

I would like to express my gratitude to all the teachers that I had the honor to learn from during this journey. A complete set of learning experiences in courses of automotive, electrical, civil and industrial engineering have been invaluable resources to expand my view of what the future of mobility will look like.

My special gratitude to Dr. Pierluigi Pisu, my academic advisor, for all his support, time, challenging questions, and guidance, that helped me so much in shaping this research.

Dr. Joachim Taiber has been there from the beginning. His vitality and passion made him the perfect mentor.

Special thanks also to Dr. Mashrur Chowdhury and Dr. Yunyi Jia for their invaluable feedback during this process.

The seed of this dissertation started with a really successful team work. Team CUICAR Tigers: I cannot thank you enough for that time and our friendship. In this regards, I am also thankful to Dr. David Smith for his support during that process.

I want to express my gratitude to all the friends I made during this time (so many of them that I rather not to personalize here). You made my life here a little bit more enjoyable during these years.

Finally I would like to thank to my family and friends from Spain for their unconditional support and the excitement that they demonstrated to me during this fulfilling experience....Gracias!

TABLE OF CONTENTS

	Page
I. CHAPTER ONE. INTRODUCTION	1
A. Objective	1
B. Motivation	2
C. Research Contributions	5
D. Broader Impacts	7
E. Research scope	8
F. Dissertation organization.....	8
II. CHAPTER TWO. AUTONOMOUS VEHICLES BACKGROUND	11
A. Levels of Vehicle Automation	11
B. Autonomous Vehicles Societal Challenges.....	12
C. Autonomous Vehicles Approaches. The Interaction Flexibility vs. User Control/Spontaneity dilemma.	14
1. OEMs approach.....	14
2. Waymo approach.....	15
III. CHAPTER THREE. LITERATURE REVIEW	18
A. Secondary Functions Control	19
1. Device Control	19
2. Situated Awareness Interactions	21
B. Primary Functions Control	22
1. Autonomous cars control	23
2. Mobile robots guidance	24
C. Dialog feedback systems	25
D. Finite State Machines (FSM) for autonomous vehicles	26
IV. CHAPTER FOUR: SYSTEM DESIGN	27
A. Flow diagram.....	27
B. Interaction command structure.....	30

Table of Contents (Continued)	Page
C. System Architecture	32
D. Simulation Framework	37
1. 3D virtual environment	37
2. System processing environment.....	37
V. CHAPTER FIVE: POINT OF INTEREST COMPUTATION	40
A. Introduction	40
B. POI computation without using external sensor information.....	42
1. Method	42
2. Simulation Results.....	48
3. Real world data collection results	51
4. Comparison simulation and real world results	70
C. POI computation with external sensor information	71
1. Method	71
2. Simulation Results.....	73
D. POI information retrieval	74
E. Summary	75
VI. CHAPTER SIX: THE ROLE OF VOICE TO CREATE THE COMMAND SEMANTICS AND DIALOG FEEDBACK TO THE USER	78
A. Introduction	78
B. Command structure implementation	80
1. Proof of concept	80
2. Voice Commands in simulation framework.....	87
C. Dialog feedback system	94
1. Voice command not recognized.....	95
2. Voice command recognized	95

Table of Contents (Continued)	Page
3. POI disambiguation.....	96
4. Command confirmation request	98
5. Command confirmation / rejection / correction by the user.....	99
6. Feasibility analysis feedback.....	100
D. Additional system modules	101
1. Path following module	101
2. Path update module	105
E. Results	107
F. Summary	109
VII. CHAPTER SEVEN: LEVEL STRUCTURED STATE MACHINE TO HANDLE CONFLICTING SITUATIONS	111
A. Introduction	111
B. States design approach	112
1. Level 1: System states	113
2. Level 2: Voice Recognition states.....	115
3. Level 3: POI computation states	116
4. Level 4: Command States	119
C. Feasibility Analysis Module.....	122
D. POI disambiguation process for Research Question 1	125
1. POI disambiguation based on map database POI property	126
2. POI disambiguation based on spatial information	128
E. Summary	131
VIII.CHAPTER EIGHT: CONCLUSIONS AND FUTURE WORK.....	132
A. Conclusions	132
B. Future Work	135

Table of Contents (Continued)	Page
IX. APPENDIX: EXAMPLES OF SIMULATION OUTPUTS.....	137
A. Command not feasible in terms of traffic rules compliance	137
B. POI disambiguation.....	138
C. Correction of POI property: color	141
D. Update command task and POI property (color)	143
X. LIST OF PUBLICATIONS	145
XI. REFERENCES	147

LIST OF TABLES

Table	Page
Table 1. SAE Levels of Vehicle Automation [23].....	12
Table 2. POI selection example.	46
Table 3. POI identification. Simulation accuracy results.	50
Table 4. POI identification. Real world accuracy results.	68
Table 5. Example of command structure with POI color information.	75
Table 6. Example of command structure.	93
Table 7. POI disambiguation. Simulation accuracy results.	127
Table 8. Map Elements database table.....	128
Table 9. POI disambiguation. Real world experiments accuracy results.	130

LIST OF FIGURES

Figure	Page
Figure 1. Voice and pointing gesture system for autonomous cars.	4
Figure 2. The interaction flexibility vs user control/spontaneous decisions dilemma.....	16
Figure 3. Human Vehicle Interaction classification based on controlled functions.	19
Figure 4. System flow diagram.	28
Figure 5. Command structure.	30
Figure 6. The proposed system as a middleware in the block diagram of autonomous vehicles.	33
Figure 7. System architecture.	34
Figure 8. Simulation framework set up.....	39
Figure 9. Dense scenario where traditional ray-casting fails.	41
Figure 10. POI computation method based on pointing ray intersection.	43
Figure 11. POI selection example map.	46
Figure 12. Pointing vectors example.	47
Figure 13. Pointing ray calculation for each vehicle position.	47
Figure 14. POI computation methods comparison.	50
Figure 15. Microsoft Kinect Sensor v2 and Tracked Body Joints [126].	52
Figure 16. GPS receiver.	52
Figure 17. Pointing Application building blocks.	53
Figure 18. File structure for the pointing application.	54

List of Figures (Continued)	Page
Figure 19. User interface of the pointing application.	54
Figure 20. Satellite view of the data collection scenarios.....	56
Figure 21. a) Experiment set up. b) Detailed view.	57
Figure 22. Coordinates transformation [132].....	58
Figure 23. a) Original sequence. b) Same coordinates filtered. c) c-means loop 1. d) c-means loop 2.	60
Figure 24. Body parts joints positions meaning.....	61
Figure 25. Pointing rays for each vehicle position (pointing right case).	62
Figure 26. POI calculation as a result of pointing rays intersections.....	63
Figure 27. POI modeled as a dynamic object with respect to the coordinate system of the car.....	64
Figure 28. EKF results.	67
Figure 29. EKF results. Change in measurements deviations.	67
Figure 30. POI computation ambiguity.	69
Figure 31. Illustration of POI computation with external sensor data.	72
Figure 32. Implemented modules in the proof of concept.	81
Figure 33. Proof of concept set up.	82
Figure 34. Proof of concept sequence diagram.....	87
Figure 35. User command grammar overview.	90
Figure 36. Trigger part of user command grammar.....	90
Figure 37. Task part of user command grammar.....	90

List of Figures (Continued)	Page
Figure 38. Location part of user command grammar.	90
Figure 39. Object part of user command grammar.	90
Figure 40. Speech recognition process example.	91
Figure 41. Colors grammar.	91
Figure 42. User confirmation/rejection grammar overview.	92
Figure 43. Rejection part of the confirmation/ rejection grammar.	92
Figure 44. Dialog feedback creation.	94
Figure 45. Command recognized feedback.	96
Figure 46. POI disambiguation dialog.	97
Figure 47. Command confirmation feedback.	98
Figure 48. Command confirmation, rejection, and correction dialog.	99
Figure 49. Feasibility analysis feedback.	101
Figure 50. Path following overview.	102
Figure 51. Simplified Ackermann model.	104
Figure 52. Examples of path updates: a) 'Go there' command; b) 'Turn there' command	106
Figure 53. New path landmarks creation depending on command semantics.	106
Figure 54. System design as a level structured state machine.	113
Figure 55. Level 1: System states.	115
Figure 56. Level 2: Voice Recognition States.	116
Figure 57. Level 3: POI computation states.	119

List of Figures (Continued)	Page
Figure 58. Level 4: Command States.....	121
Figure 59. Command confirmation / rejection / update process.	121
Figure 60. Feasibility analysis modules integrated in the system process.....	122
Figure 61. Map database tables.....	123
Figure 62. Feasibility analysis module.	125
Figure 63. POI disambiguation process by map element name/property.	126
Figure 64. Result of POI disambiguation by map element name/ property.....	127
Figure 65. POI disambiguation process by spatial relation of map elements.	129
Figure 66. Map elements spatial relations based on alpha angle.	129
Figure 67. Result of POI disambiguation by spatial relation.....	130
Figure 68. Example of command not feasible.	137
Figure 69. Resulting command and path for STOP command.	138
Figure 70. POI disambiguation dialog.	139
Figure 71. Result of POI disambiguation.	140
Figure 72. Correction of POI color dialog.	141
Figure 73. Color filtering to determine the new POI coordinates.....	142
Figure 74. Result of POI color update.	142
Figure 75. Dialog for task and color update.	143
Figure 76. Result from task and color update.	144

I. CHAPTER ONE. INTRODUCTION

A. Objective

The **general objective** of this research is *to develop a strategy for a multimodal human vehicle interaction system based on voice and pointing gesture commands that enables users of autonomous cars to make and communicate situation awareness spontaneous decisions to the vehicle, with the objective of commanding it towards a target point of interest; modifying, under feasible safe conditions, its pre-defined route.*

This strategy has the objective to solve the interaction flexibility vs. user control/spontaneous decisions dilemma of autonomous cars. In this way, it is expected that this research will help to increase the current user acceptance levels of autonomous vehicles.

In particular, this dissertation is focused on answering three fundamental research questions:

Research Question 1 (RQ 1): *How can the POI computation be improved for situated awareness interactions, especially in cluttered environments?*

Research Question 2 (RQ 2): *How can voice and pointing gestures be combined to form a semantically complete command to the autonomous vehicle and give the proper feedback for conflicting situations such as POI identification ambiguity or not feasible user command?*

Research Question 3 (RQ 3): *How can this system be designed so that it can handle conflicting situations such as feasibility of the command or POI ambiguity?*

B. Motivation

Autonomous cars is a topic of broad and current interest in the automotive industry. Some forms of automated vehicle technologies such as Advanced Driver Assistance Systems (ADAS) are already being introduced in production vehicles [1]. Vehicle manufacturers are announcing their plans to gradually introduce automation capabilities in their vehicles [2], [3]. Others, such as Tesla, are already offering autopilot functions in their production cars, by means of a simple over-the-air software update [4]. However, the most promising developments will take place between the years 2025 and 2035, when fully autonomous cars are expected to be on the roads [2], [5]-[7]. Distinguished members of the Institute of Electrical and Electronics Engineers (IEEE) have selected autonomous vehicles as the most promising form of intelligent transportation, anticipating that by 2040 fully autonomous cars will represent up to 75% of the cars on the road [8]. These cars will improve road traffic and safety, as they will be designed to monitor the roadway conditions for an entire trip [9]. In addition, people that currently cannot drive will be able to enjoy the freedom of mobility [10].

A study conducted by IEEE that surveyed more than 200 experts from different areas of automotive engineering, showed that by 2035 most of the new autonomous cars will not have steering wheels or acceleration and brake pedals [11]. The same direction is

followed by Waymo¹ [12] with its autonomous vehicle prototype, which originally did not have any driver controls [10], [13]-[15]. By removing the steering wheel from the vehicle, the driver will be able to sit anywhere in it. This will change the interactions inside the car and could lead to redefine the vehicle interior standards. Instead of having restricted interaction areas as known today [16], the car as a whole can be transformed to an entire space that provides interaction flexibility for the user.

However, in order to achieve the needed user acceptance levels for the adoption of these new autonomous vehicle concepts, some important human-vehicle interaction challenges should be addressed. One of the most important is related with the lack of user control over the car [8], [17], [18]. On the one hand, delegating all the decision power to the vehicle could be perceived as unfamiliar to current drivers, affecting to the aforementioned user acceptance of autonomous vehicles, which is currently around 42% [19]. On the other hand, there will always be some scenarios related to localized or situated awareness at runtime, where users will make on-route spontaneous decisions, such as taking a detour or stopping at a specific location not pre-defined in the original route, for which the driver will require some “command” over the vehicle. Current vehicle concepts do not allow to perform these functions in a relatively short period of time, unless the user is in front of the steering wheel; which is the approach taken by vehicle manufacturers [20]-[22] and affects the potential interaction flexibility in the vehicle. In addition, the aforementioned Waymo approach does not allow the user to have some “command” over the autonomous car, potentially affecting the user acceptance

¹ The Google Self-driving car project recently became a new company called Waymo. For the purpose of this dissertation the terms ‘Waymo approach’ and ‘Google approach’ should be considered equivalent.

issue related with the lack of user control. Thus, there is a gap in the development of a natural human vehicle interaction system that allows the user of an autonomous vehicle to give instructions to it regardless of where he is seated. This system should provide to the user enough feeling of control, while maintaining the interaction flexibility inside the vehicle. This would help to increase the user acceptance levels of this technology.

This research proposes the use of a multimodal interaction system based on voice and pointing gesture commands that enables designated users to make and communicate spontaneous decisions to the autonomous car, modifying, under feasible safe conditions, the pre-defined autonomous route on real-time. For example, similar to giving directions to a taxi driver, a user will be able to tell the car «*Stop there*» or «*Take that exit*». In this way, the user control/spontaneity vs the interaction flexibility dilemma can be solved.

Figure 1 shows an illustration of the proposed human vehicle interaction system.

The strategies developed in this dissertation allow the vehicle intelligence to interpret the semantics of the command, identify the relevant point of interest to update



Figure 1. Voice and pointing gesture system for autonomous cars.

its route, and execute the corresponding task. In addition, a dialog based feedback system is developed in order solve potential conflicting situations such as command feasibility analysis, POI computation uncertainty or the command misunderstanding.

C. **Research Contributions**

This dissertation develops a strategy to enable multimodal human vehicle interactions based on voice and pointing gesture commands to allow users of autonomous cars to make situation awareness spontaneous decisions and communicate them to the vehicle. In this way, under feasible safe conditions, the vehicle intelligence can execute the user's desired command and update its predefined autonomous route according to it. In particular, taking into account the research opportunities resulting from the background and literature review presented in Chapters II and III, the main contributions of this dissertations are:

- 1) Implementation of a POI computation strategy for situated awareness scenarios using pointing gestures. Of special interest is the POI computation in cluttered and dense environments, where more than one POI may intersect the pointing vector in a 2D Map. The results from the proposed POI computation method based on pointing rays intersection shows an increase of POI identification accuracy of 36.25% when compared with the traditional ray-casting method.
- 2) Combination of voice and pointing gestures in semantically complete commands that can be used to modify the pre-defined route of the autonomous

car. In this way, depending on the combination of the command semantics and POI computation, the route of the autonomous car can be properly updated.

This is demonstrated by implementing the simulation framework, described in Chapter IV.

- 3) Development of a dialog feedback system that solves and make the user aware of potential conflicting scenarios, such as POI computation ambiguity, misunderstanding of the user command or negative result of the command feasibility analysis. This dialog feedback system allowed to solve all the POI identification errors of Research Question 1. In addition, it allows the user to be aware of the system's understanding and state.
- 4) Design and development of the system based on a level structured Finite State Machines (FSM) approach that allows to analyze the feasibility of the given command in terms of safety and traffic rules compliance, as well as to handle the previously mentioned challenging situations such as POI ambiguity or command misunderstanding. This scheme allows to successfully implement the system of this dissertation as well as to solve ambiguous situations such as the aforementioned POI computation ambiguity. In addition, this design approach allows the system to be scalable and modular, in a way that other interaction modes would be able to reuse the same system design structure maintaining its main modules and states.

The application of these contributions constitute the foundations of the human vehicle interaction system for autonomous cars presented in this dissertation.

D. **Broader Impacts**

The research presented in this dissertation provides the foundations of a natural and flexible multimodal interaction system for users of autonomous cars. This system allows them to command the vehicle when needed or wanted; thus, enabling flexible interactions inside the car while maintaining the feeling of control over it. By providing such user control, it is expected an increase of the current user acceptance levels of this technology.

The outcomes of these research can also be applied to other human vehicle interaction scenarios. Such is the case of situated awareness scenarios where the user wants to gather information from the vehicle surroundings, which can be projected in the heads up display of the windshield or windows of a car. Using augmented reality technology, the user can receive information of the target POI in the surroundings of the vehicle, such as nearby restaurants, and get new services such as hours of operation information, reservations, digital coupons or others.

In addition, this research can also be applied to the field of teleoperation of mobile robots, including cars. Voice and pointing gesture commands can replace the commonly used joystick in teleoperations, providing a more natural and flexible interaction technique in this field.

Finally, this research can be scaled to a wider research and discussion of how humans and autonomous vehicles can effectively communicate and collaborate, not only by means of voice and pointing gestures, but by using other interaction technologies and human behavior recognition techniques, from the inside and outside of the vehicle. In this way, vehicle and humans can be aware of each other state and be able to collaborate towards the goal of creating a safer and more comfortable intelligent driving experience.

E. **Research scope**

This research is limited to the implementation of the main modules of the proposed interaction system, which are described in Figure 7 of Chapter IV. It is assumed that the vehicles are SAE Level 4 or 5 capable (i.e. fully autonomous) and equipped with the needed sensors and other modules needed for the vehicle automation, such as path planning, behavioral planning, localization, situation awareness, or others. In addition, it is assumed that the vehicle is equipped with the needed hardware for enabling voice and gesture recognition, as well as heads up display technology (HUD) in the windshield of the car, so that the intersection of the pointing vector and the windshield could be highlighted (similar to the mouse pointers of a standard computer). An illustration of this can be seen in Figure 1 of this chapter.

F. **Dissertation organization**

The rest of this dissertation is organized as follows. Chapter II describes a high level autonomous vehicle background. This description leads to the identification of the

interaction flexibility vs. user control/spontaneity dilemma that this dissertation intends to solve. As it was mentioned in this chapter, the proposed solution for this dilemma is based on the development of a voice and pointing gesture based human vehicle interaction system. Thus, Chapter III presents the literature review related with human vehicle interaction systems, as well as an overview of dialog feedback systems and finite state machines applied to autonomous cars.

Chapter IV presents an overview of the main building blocks of the system developed in this dissertation, as well as the simulation framework developed to demonstrate its main functions.

Chapter V focuses on answering *Research Question 1* related with the target POI computation. The first part of the chapter proposes a method to compute the POI when it is out of the vehicle sensor range. This method is based on pointing rays intersection. The second part of this chapter introduces depth information from the vehicle sensors.

Research Question 2 is the focus of Chapter VI. By combining voice and pointing gestures in the same command structure, the pre-defined autonomous route of the vehicle can be updated. In addition, this chapter analyzes the role of a voice based dialog feedback system to make the user aware of the system status as well as to collaborate and solve potential challenging situations, such as POI ambiguity. This is demonstrated using the simulation framework described in Chapter IV.

The system developed in this dissertation is designed using a framework based on level structured finite state machines (FSM). This is the focus of *Research Question 3*, explored in Chapter VII. The FSM structure developed in this chapter allows to keep

track of the system states at different levels, analyze the feasibility of the command, and provide the necessary system outputs for the dialog feedback system described in Chapter VI.

Finally, Chapter VIII describes the main conclusions of this dissertation as well as potential future work and directions that may result from this research.

II. CHAPTER TWO. AUTONOMOUS VEHICLES BACKGROUND

A. Levels of Vehicle Automation

The SAE (Society of Automotive Engineers) defines in its Information Report J3016 [23] five different levels of vehicle automation (Table 1), depending on what entity (the system-car, or the human-driver) performs the primary functions of steering and accelerating, and monitoring of the driving environment.

In this way, Levels 0, 1 and 2 need constant human monitoring of the driving environment; while in Levels 3, 4 and 5 the automated car monitors it; being Levels 4 and 5 the ones corresponding to a fully autonomous driving mode, without requiring any user intervention.

Currently many production vehicles and prototypes are capable of driving with some levels of automation. Lane departure warning, parking assist, adaptive cruise control or lane keeping assist are some of the Level 2 technologies that have been demonstrated. In addition, the Tesla Autopilot has already been launched [4]. We can consider this a Level 2 system. However, probably one of the most disruptive advancements towards fully autonomous cars is being developed by Waymo, with its ambition to remove completely the users out of the driving equation [10].

While it is clear that the path towards fully autonomous cars is on its way, some major societal challenges must be addressed to make this technology a reality. Next subsection discusses some of them.

Table 1. SAE Levels of Vehicle Automation [23]

Level	Description
Human driver monitors the driving environment	
0 – No Automation	The full-time performance by the human driver of all aspects of the dynamic driving task, even when enhanced by warning or intervention systems
1 – Driver Assistance	The driving mode-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the human driver perform all remaining aspects of the dynamic driving task
2 – Partial Automation	The driving mode-specific execution by one or more driver assistance systems of both steering and acceleration/deceleration using information about the driving environment and with the expectation that the human driver perform all remaining aspects of the dynamic driving task
Automated driving system monitors the driving environment	
3- Conditional Automation	The driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task with the expectation that the driver will respond appropriately to a request to intervene
4 – High Automation	The driving mode-specific performance by an automate driving system of all aspects of the dynamic driving task, even if a human driver does not respond appropriately to a request to intervene
5 – Full Automation	The full-time performance by an automated driving system of all aspects of the dynamic driving task under all roadway and environmental conditions that can be managed by a human driver

B. Autonomous Vehicles Societal Challenges

Several studies have analyzed some of the societal challenges that autonomous vehicles will face and need to be solved for their adoption [2], [6], [7], [20], [24]-[30].

Three of the most relevant being:

- 1) ***Regulatory changes*** should create the proper environment to allow autonomous vehicle testing and operation in public roads. Some states of the US, as well as some European countries allow autonomous cars to be tested in public roads for research purposes [24], [25]. However, a driver must be

inside the vehicle at all times, either in immediate physical control of the vehicle or actively monitoring the vehicle's operation. In this way, the driver must be capable of taking over immediate control of the car. This means that the car must be equipped with steering wheel and accelerator and brake pedals [31]. While being a safety critical rule, this could slow down the development and testing of innovative autonomous vehicle technologies [32]. Although a recent statement of the National Highway Traffic Safety Administration (NHTSA) states that the computer inside Google's self-driving car can be considered the driver of the vehicle (thus, potentially redefining the previously mentioned vehicles' control elements requirements) [33], this regulatory discussion is still a challenging open topic [26], [34].

- 2) *New liability models* should be created by insurance companies in order to incorporate autonomous cars into their policies [2], [6], [7], [28]-[30].
- 3) As it was stated in the Chapter I, there are some concerns related to the *user acceptance levels* of this technology. It seems to be a gap between the projected number of autonomous vehicles in 2040 (around 75%) [8] and the user intention for purchasing/leasing one (between 24 and 39%) [35], [36]. In addition, a recent survey by AAA reported that 75% of Americans would be afraid to ride in an autonomous vehicle [37]. Users may not want to delegate all the control and decision making process to the car [8], [17], [18], [20].

Increasing the user acceptance levels of autonomous vehicles is the motivation of the system developed in this research. By using voice and pointing gesture commands, the human vehicle interaction approach presented enables the users of an autonomous car the ability of taking control of it when they need or want it, while maintaining the ride safety conditions and the interaction flexibility that new vehicle designs and trends are proposing [38], [39].

C. Autonomous Vehicles Approaches. The Interaction Flexibility vs. User Control/Spontaneity dilemma.

Currently there are two main approaches in the development of autonomous vehicles: the vehicle manufacturers (OEMs) approach and the Waymo approach.

This subsection explores each of them and analyzes their advantages and disadvantages in terms of the capability of the car to provide control to the user and interaction flexibility inside it.

1. OEMs approach

Vehicle OEMs have plans to develop autonomous cars. However, they only expect to reach Level 3 of automation in the near term. Moreover, they consider that the self-driving mode would be implemented in the boring parts of driving [21], such as traffic jams, highway driving, etc. However, the driver still has to be attentive during them.

This approach maintains the steering wheel and accelerator and brake pedals in the vehicle design as elements to allow the driver to take control over the car if needed or wanted. This could be seen as an advantage in terms of improving the aforementioned user acceptance levels of autonomous vehicles. However, it prevents meeting all the expectations that are being generated around innovative ways of interacting with autonomous cars [38], [39]. By maintaining the steering wheel, drivers have limited interaction flexibility, as they will always have to be in the drivers' seat or in a seat where they could reach these elements immediately, monitoring the environment and being prepared in case they need or want to take the control over the car. In this scenario, all the innovation potential that fully autonomous cars can enable will be affected by the constraint of having the steering wheel in them.

2. Waymo approach

Although Waymo has been forced to add steering wheel and accelerator and brake pedals to its prototype vehicles in order to comply with the regulations related with autonomous vehicles testing in public roads [40], its final goal is to remove these elements [10], [13]-[15]. In this way, Waymo is targeting Levels 4 and 5 of vehicle automation, where human intervention (and thus, human driving mistakes) is minimized during the driving process [10]. In addition, as it was mentioned previously, the NHTSA has recently accepted to consider the computer of the Waymo's car as the driver of the vehicle [33]. This could help to accelerate the introduction of new vehicle models without

steering wheel and accelerator and brake pedals; potentially redefining vehicle interior designs and human-vehicle interaction possibilities.

While this approach provides the user a flexible interaction space [15], it prevents him to make and communicate in real-time spontaneous decisions over the autonomous route. A recent patent of the company indicates that the only user controls available inside the vehicle are limited to ‘start’, ‘emergency stop’ and ‘pull over’ buttons [13]. This could affect negatively to the users’ feeling of control over the driving process; thus, influencing the potential user acceptance on this technology.



Figure 2. The interaction flexibility vs user control/spontaneous decisions dilemma.

As it is illustrated in Figure 2, none of the two main autonomous vehicle approaches described in this chapter provides an autonomous vehicle concept that enables user control and spontaneous decision making over the autonomous route, while maintaining the interaction flexibility. Solving this ‘interaction flexibility vs user control/spontaneous decisions dilemma’ is the focus of this research.

III. CHAPTER THREE. LITERATURE REVIEW

This research presents the use of human machine interaction technologies to provide users of autonomous cars control over them. In particular, voice and pointing gesture based interactions.

On the one hand, human vehicle interaction technologies have been traditionally proposed as a way to interact with the secondary functions of the car; that is, those that are not directly related with the driving tasks [16].

On the other hand, some literature propose the use of human interaction technologies to control the primary functions (those related with the driving task) of mobile robots (autonomous cars included).

This chapter gives an overview of the aforementioned use of human interaction techniques; following the classification shown in Figure 3.

In addition, some background of two important building blocks of these research, such as the use of finite state machines (FSM) to perform users' command feasibility analysis and vehicle-human feedback systems is provided.

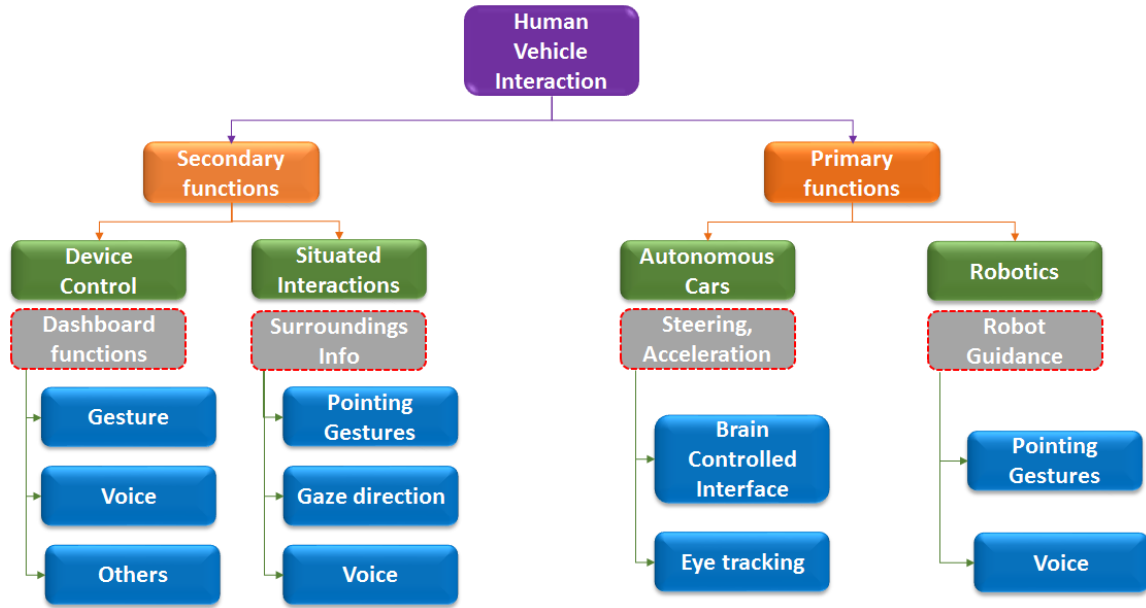


Figure 3. Human Vehicle Interaction classification based on controlled functions.

A. Secondary Functions Control

Secondary functions are those that are not directly related with the driving task [16]. Such is the case of the use of the vehicle infotainment functions and other vehicle controls such as the A/C, windows or others. Recently, other type of secondary functions, called situated interactions, where users gather information of the vehicle surroundings, have been proposed [41]-[45]. For both cases, different human vehicle interactions techniques are found in the literature.

1. Device Control

Human vehicle interactions with the secondary functions of the vehicle that involves some device control, such as interacting with the center console of the car, can

cause driver distraction [46]. For this reason, research on different human vehicle interaction technologies to minimize drivers' distraction when interacting with these functions is found in the literature [47]-[55]. In addition, some vehicle manufacturers are already offering these technologies [56]-[60].

Gesture control is identified as a natural, flexible and innovative way to interact with the secondary functions of the car [61]-[63]. Its main benefit is that the user does not have to physically touch the device he intends to control; thus keeping his eyes on the road and hands on the steering wheel. This technique can be based on different sensor technologies; such as vision [41], [62]-[65], electric field [66]-[68] or electromyography sensors [69], [70]. In this way, features of the gestures can be extracted and compared with a predefined gesture vocabulary. Using machine learning techniques the corresponding function of the gesture can be recognized [71]. Although some research shows that the definition of gesture vocabularies is a challenging task [72]-[75], the benefits of gesture interactions to reduce driver distractions have been recognized [47], [76]. In fact, some vehicle manufacturers and other stakeholders are proposing cars equipped with this technology [52]-[54], [77].

Voice recognition systems and their effects on driver distraction is also widely found in the literature [78]-[82]. The advancements on this field allowed to develop voice recognition systems in such a way that the communication between human and vehicle can be performed in a natural and flexible manner [78], [83]-[86]. Vehicle manufacturers are already offering this interaction technology in their production vehicles [57]-[60].

2. Situated Awareness Interactions

These interactions are related with the query of information of points of interests in the vehicle surroundings, such as buildings, restaurants or others. As it has been proposed by Mercedes on its futuristic Mercedes DICE concept [87] and by BMW [41], in situated awareness interactions, the user can interact through the windshield of the car to gather vehicle's surroundings information. For this, the identification of the target POI is needed.

GM [88] proposed a method to calculate the coordinates of the point of interest by means of pointing gestures. In this way, using the direction of the pointing vector, the vehicle orientation and position, and a POI database, the desired POI can be calculated. This approach is based on the pointing technique called ray-casting, traditionally used in the field of virtual reality [89]-[91].

In [42] experiments were performed in static simulation lab conditions to calculate the location of the target by means of multi-person pointing vector intersections. In order to calculate the target POI coordinates, the authors assumed that two persons in the car were pointing to the same location. The authors stated that their method is applicable for target POIs at distances less than 40m. The results showed the errors increased with the distance; being the average error of a POI at 10 meters from the car, 2.59 ± 1.17 meters.

Face and eye tracking have also been proposed for calculating coordinates of POIs in the vehicle surroundings [43]. Similar to the case of pointing gestures, the gaze direction is combined with the vehicle heading and position to trace the pointing ray.

This ray is used to find its intersection with a list of POI stored in an annotated map database. The first POI or POI area that intersects the pointing ray, is considered to be the target POI. The experiments in real driving scenarios achieved an overall accuracy rate up to 65%.

In order to support the POI calculation using gaze direction, research done by Honda R&D [44], [45], proposed the use of head orientation (gazing direction) and users' queries (using voice recognition) to calculate and provide information of the target POI. In this way, linguistic cues such as spatial references (left or right) or business category (restaurant, shop, or others) were used to support the target POI identification. In addition, the head orientation, the angle between the gazing ray and the locations of the POIs stored in an annotated map database, were used to identify the most likely target POI. Using this method the authors reported a success rate of 67.2% in two driving scenarios.

B. Primary Functions Control

Primary functions are those that are directly related with the driving task [16]. Such is the case of accelerating, braking and steering. Although limited, some research has been done related to the use of human interaction technologies to control these functions in autonomous cars [92], [93].

In this research, the control of the movement of mobile robots is also considered under the primary functions umbrella. In this regards, human machine interaction techniques such as voice and gestures are also used to guide mobile robots [94]-[96].

The following subsections explore the aforementioned research works.

1. Autonomous cars control

The control of autonomous vehicle primary functions by means of innovative human vehicle interactions has been explored by a team of Free University of Berlin in Germany.

The use of eye trackers for controlling the steering of the autonomous vehicle “Spirit of Berlin” was proposed in [92]. In this way, the user eye movements were mapped to steering angles, which were applied to steering commands. The authors of this research found that the fluctuations of the eyes positions caused non-smooth steering commands. As a result, the operation of the vehicle via eye tracking could be exhausting over the time. For this reason, limited applications in key areas, such as turning left or right in intersections, were proposed as future use cases of this technology.

The use of Brain Computer Interfaces (BCI) as a way to control the steering, acceleration and braking of the autonomous vehicle “Made in Germany” was explored in [93]. The experiments for this research were performed while the vehicle was following a predefined route. In this way, the test subject was asked to try to keep the vehicle centered in the predefined route’s lane. The researchers of this work recognized the difficulty for a human to estimate his distance to the center of the lane to minimize the lateral error. For this reason, the driver was looking at a computer monitor inside the vehicle instead of the exterior of the car. When the test person had to control only the steering of the vehicle at 2 m/s, the standard deviation of the lateral error was

1.875±0.2m, while the one for the orientation was 0.20 rad. These errors increased to 4.484 m. and 0.222 rad. for a speed of 5 m/s. When the test person had to control acceleration, braking and steering, the standard deviation of the lateral error was 2.765 m. and the orientation error was 0.410 rad. When the test person had to decide in certain key areas, such as intersections, the direction to follow (left or right), 90% of accuracy was achieved.

2. Mobile robots guidance

The POI identification with the objective of guiding mobile robots have been proposed in several research studies [94]-[97].

A method for target point estimation from a pointing gesture by means of a low-cost monocular camera mounted on a mobile robot and a multilayer perceptron neural network is presented in [94]. In this way, the distance and angle that the robot should follow was calculated. The results of this research showed that 82% of the trials has average position errors in a range of 31-59 centimeters, which are high taking into account that the potential target markers were in a range of 1-3 meters from the robot.

The computation of the pointing direction using a Time of Flight (ToF) camera and a Gaussian Regression Process is proposed in [95]. In this case, the distance error was 0.17±0.12m; and the angular error was 2.79±1.99degrees.

Multimodal interactions based on voice and pointing gestures are proposed in [96] and [97] to command an assistive robot. In these research works, the human operator can command the robot to perform certain tasks using a natural dialog interface based on a

voice and pointing gesture command structure and a dialog feedback system. In this way, if more information is needed to perform the task a dialog was established between human and robot, until no ambiguity is detected.

C. **Dialog feedback systems**

The previous subsection showed how dialog and voice recognition systems can support human machine interactions. This has been also confirmed by different research studies outside the automotive or robotics field [98]-[100], and demonstrates the importance of designing proper feedback dialog systems to provide a natural human-machine interaction experience.

In this regards, a natural language system for robotics operations was implemented in [101]. Via a collaborative dialog between the human operator and the robotic system, new tasks and actions were taught to the robot.

For the particular case of automotive applications, research shows that an in-vehicle dialog systems have to be designed in a way that the cognitive load of the driver is not affected [102]-[104].

As it was described in Chapter I, this research implements a command structure based on voice and pointing gestures to provide semantically complete commands, and a feedback system to solve potential conflicting scenarios, such as POI ambiguity or negative result of the command feasibility analysis.

D. Finite State Machines (FSM) for autonomous vehicles

Finite states machines have been widely used in autonomous vehicle applications. Participants of the DARPA Challenge used them to define the behavior of their autonomous cars in different traffic situations, where different traffic rules apply. Such is the case of intersection crossing, parking navigation or stop sign wait. This allowed the DARPA Challenge vehicles to implement decision making tasks in real time [9], [105]-[108]. Other research teams also integrated FSM in their models to implement human driver decisions models and perform motion planning tasks [109]-[111].

In addition, FSMs have been demonstrated to be useful in path planning scenarios with unknown and dynamic environments [112]-[114] or other applications such as vehicle platooning [115], [116].

As it was mentioned in Chapter I, the system presented in this research is based on level structured state machines, so that the system states can be tracked and can handle potential challenging situations, such as POI ambiguity, command misunderstanding or not feasible command in terms of safety and traffic rules compliance. In addition, designing the system in this way allows to provide the proper dialog feedback system to the user.

IV. CHAPTER FOUR: SYSTEM DESIGN

This chapter, although does not answer directly to a research question of this dissertation, is needed to build the foundations of the system that forms the general objective of the presented research.

A. **Flow diagram**

Figure 4 shows a flow diagram describing the basic functionality of the proposed system in this dissertation.

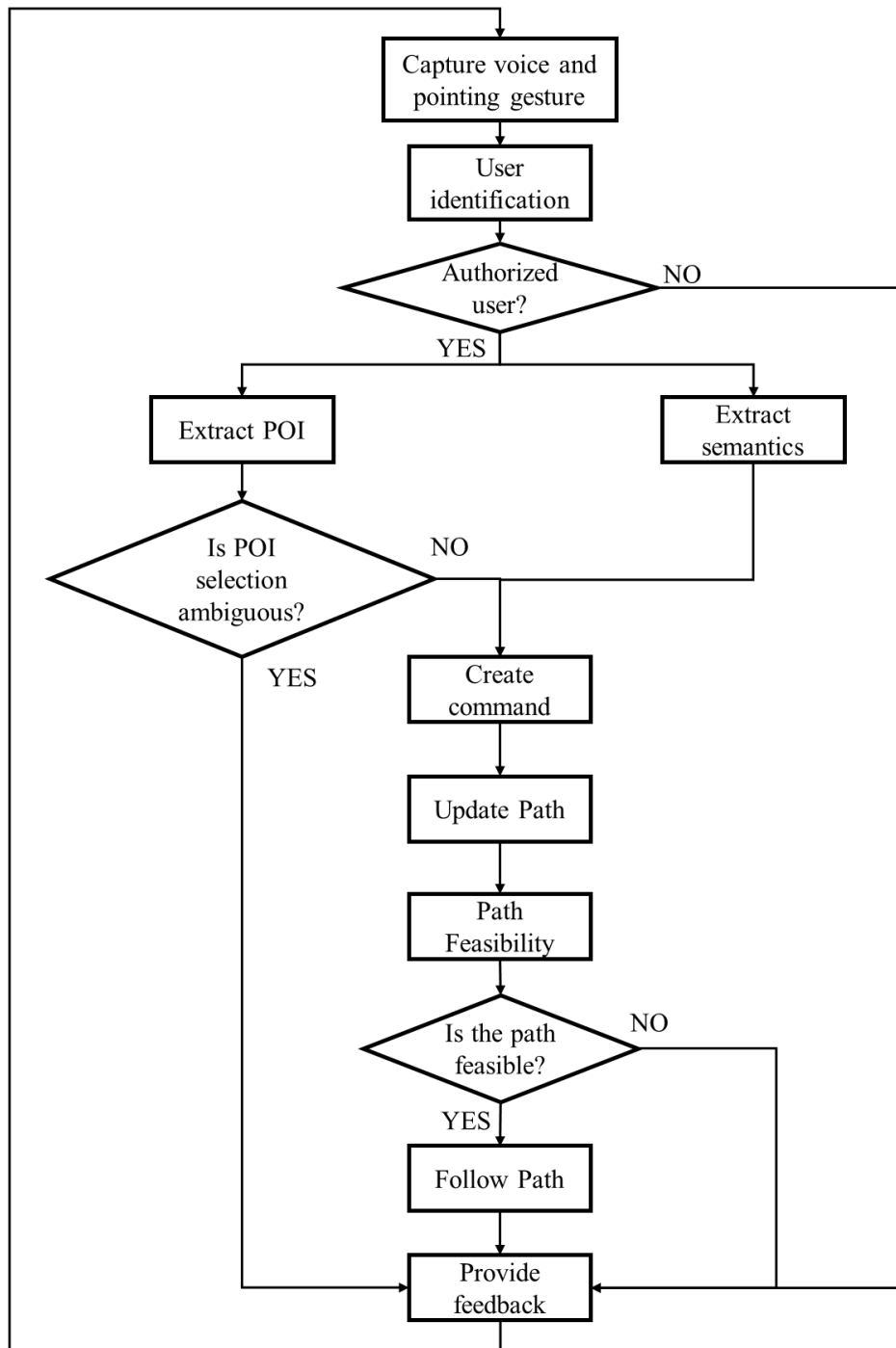


Figure 4. System flow diagram.

In order to provide a voice and pointing gesture based human-vehicle interaction system that enables users' spontaneous decisions over the autonomous route as well as interaction flexibility inside the vehicle, the system monitors the interaction space of the car as a whole. In this way, the system can capture users' voice and pointing gestures ('Capture voice and pointing gesture'). As it is described in the next subsection, a trigger command activates the voice and pointing gesture command recognition process. Although this is not the current scope of this research, the system will only execute the commands performed by those users that have 'driver' privileges. Thus, a user role identification step is needed ('User identification'). Once the command is performed by the user, the system identifies the target Point of Interest (POI) ('Extract POI') in the vehicle surroundings that the user is pointing to, and complements it with the semantics of the voice command ('Extract semantics', 'Create command'). If the result of the 'Extract POI' phase is ambiguous (more than a potential target POI was identified), the feedback system of the car informs this issue to the user in order to solve it ('Provide Feedback'). If there is no ambiguity, the command information is used to update the autonomous vehicle's pre-calculated path ('Update Path'). The feasibility of this new path is evaluated based on the driving context captured by external sensors and mapping information ('Path Feasibility'). If this feasibility analysis determines that the calculated path cannot be executed under safe conditions or does not comply with some traffic rule, the system will not execute the command and will communicate it to the user through the feedback system of the car ('Provide Feedback'). Otherwise, the command is executed and the new path is followed ('Follow Path').

For example, the command «OK Car, stop there» combined with a pointing gesture will cause the car to stop close to the target pointed location, as long as the vehicle detects that safety conditions and traffic rules are not compromised.

For Figure 4, ‘Extract POI’ will be explored in *Research Question 1*; ‘Extract semantics’, ‘Create command’, and ‘Provide feedback’ in *Research Question 2*; and ‘Path Feasibility’ and ‘Is POI selection ambiguous?’ in *Research Question 3*. Although they are not part of the fundamental research questions presented here, the phases ‘Capture voice and pointing gesture’, ‘Update path’ and ‘Follow path’ will be also implemented as part of the works needed to complete this research. In addition, it is important to note that Figure 4 is a simplified version of the system developed in this dissertation. A complete design based on FSM will be further described in Chapter VII.

B. Interaction command structure

In order to process the user commands, the system follows the structure shown in Figure 5.

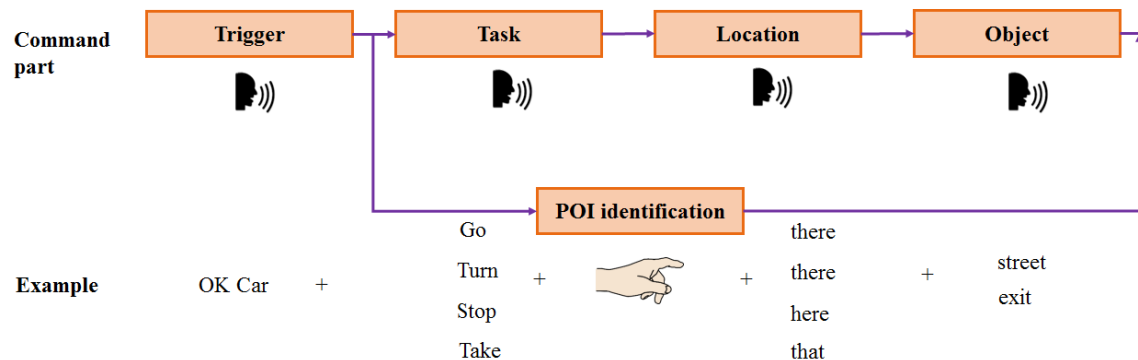


Figure 5. Command structure.

As it is depicted in Figure 5, the command structure is formed by the following parts:

- 1) **Trigger command** is a word or phrase used to activate the voice and pointing gesture command recognition process. In this way unintended commands can be avoided. The interaction mode of this command part is *voice*.
- 2) **Task** is the action that the user intends the car to perform. The interaction mode of this command part is *voice*.
- 3) **POI identification** is used to relate the instruction given with a vehicle's surroundings target POI that a user is pointing to. The information in this part can include the coordinates of the POI and other characteristics that may be available (for example in an annotated map), such as building color, type, or others. The interaction mode of this command part is *pointing gesture*.
- 4) **Location** is a word that provides 'spatial' meaning, such as *there*, *here*, or others. Although it is not explored in this dissertation, the semantics of the location could be used to determine if the point of interest is close or far. The interaction mode of this command part is *voice*.
- 5) **Object** is a word or phrase that refers to the target object. Such is the case of a street, exit, or other. Although it is not explored in this dissertation, the semantics of the object could be used to help the vehicle make more educated decisions. The interaction mode of this command part is *voice*.

To illustrate this command structure, the following examples are proposed:

- «*OK Car + Go + pointing gesture + there*» to command the car to go towards a certain location.
- «*OK car + Turn + pointing gesture + there*» to turn into a street in a city.
- «*OK car + Stop + pointing gesture + here*» to ask the car to stop at a certain location.
- «*OK car + Take + pointing gesture + that + exit*» to take an exit in a highway.

For example, in the last one: «*OK car*» is the voice activated trigger command, «*take + ... + that + exit*» is the task that the car has to perform, and the pointing gesture indicates to the car which exit it has to take. Under this structure, the pointing gesture and the voice part of the command could happen in parallel, creating a semantically complete command.

The following subsection describes the system architecture the proposed system.

C. **System Architecture**

To implement the proposed system in an autonomous car, it is necessary to relate the pointing gesture captured inside the car with its outside world by relating the pointed object with its physical location, the type of object and the context associated to it. In this way, the system can make educated decisions about the command and new calculated path. For example, if a user is giving a «*Stop*» command to the vehicle, and pointing to a world location where stopping is not allowed (for example, a road intersection), the

system will not execute the command and will inform to the user about it. This functionality is done by integrating the proposed system as a block in the commonly known autonomous vehicle building blocks.

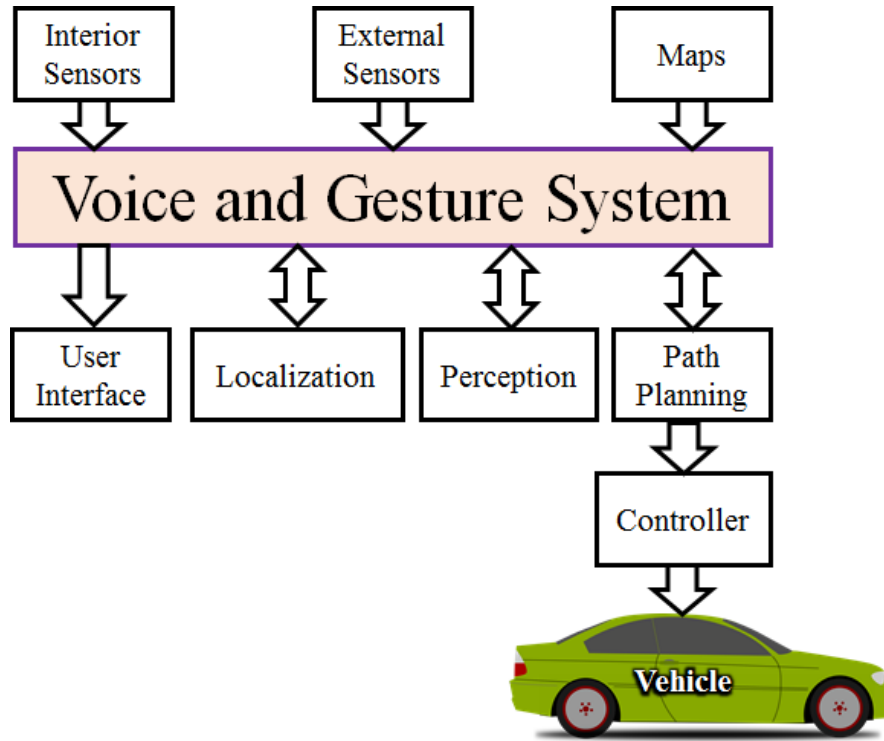


Figure 6. The proposed system as a middleware in the block diagram of autonomous vehicles.

As it is shown in Figure 6, the proposed voice and pointing gesture system acts as a middle layer between the sensing input blocks of the autonomous car and its algorithmic blocks. In this way, when a user performs a voice and pointing gesture command, the interior sensors block detects it. This information, combined to the maps block and the external sensors block is fused in order to calculate the POI which the user is referring to. The retrieved information is used by the localization, perception and path planning blocks to update the autonomous route and evaluate its feasibility in terms of

safety and traffic rules compliance. If the new path is feasible and no command ambiguity is detected, the car will follow it by means of the vehicle controller. Otherwise, the new path is discarded and the corresponding feedback is given to the user through the user interface of the vehicle, in order to communicate this issue to the user and try to solve it though more feedback steps.

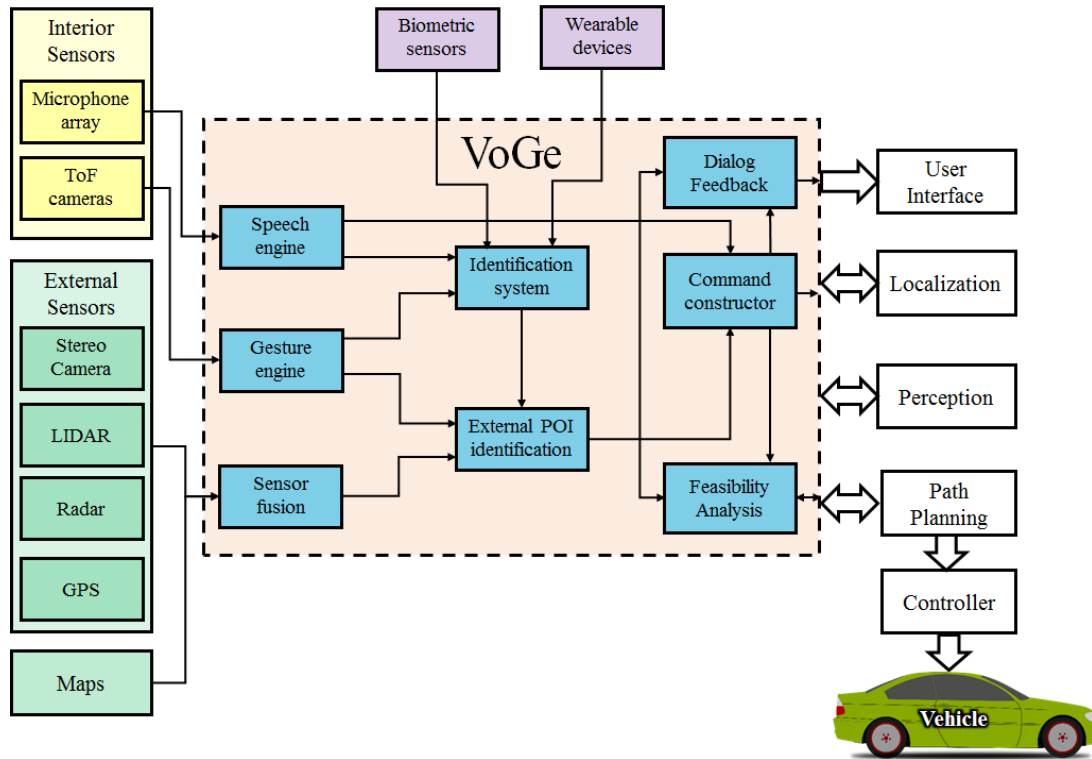


Figure 7. System architecture.

A more detailed diagram of the proposed system is depicted in Figure 7. As it was already mentioned, the voice and pointing gestures are captured by the interior sensors of the car. The voice based trigger and action parts of the command are captured by a microphone array and transmitted to the speech engine subsystem for processing. Once the trigger is processed by the speech engine, the rest of the modules wake up to start

processing the interaction information. At the same time, the pointing gesture sequence is captured by a Time of Flight (ToF) camera and sent to the gesture engine in order to process the coordinates of the body parts that will be used to form pointing vector. As it will be further explained in the Chapter V related with **Research Question 1**, this vector forms an angle with the heading direction of the vehicle, which is given by its GPS sensor. The GPS coordinates of the external POI pointed by the user is the outcome of the external POI identification module. Using the pointing ray created in the gesture engine module and the Point Cloud (PC) generated by the sensor fusion of the different vehicle external sensors (LIDAR, radar and others), the intersection point between the ray and the PC is calculated. This point is translated into GPS coordinates, forming the coordinates of the POI that the user is pointing to. As it will be explained in the Chapter V related with **Research Question 1**, part of the POI identification strategy relies only on pointing vectors' intersections and not in external sensors. This is due to the fact that the range of these sensors may be limited and the user may be pointing to a target location farther than the external sensors range.

Once the POI has been identified and the voice command recognized by the speech engine, the command constructor module creates the command data structure previously described (for example, the command «OK Car, stop there» and all its relevant information such as the POI coordinates). This is part of the **Research Question 2** presented in Chapter VI.

The output information of the generated command is used to update the path of the autonomous route. However, before executing it, it is necessary to evaluate the

feasibility of the new path. Using the command information, the new path, and the annotated map data available, the feasibility analysis module evaluates if the new path is feasible in terms of safety and traffic rules compliance. The strategy to perform this feasibility analysis is part of **Research Question 3** presented in Chapter VII.

If the result of the feasibility analysis is positive, the system will send the command to the controller module of the vehicle, so it can be executed. Otherwise, the system will not execute it. In either case, the system will communicate its result using the dialog feedback module and the user interface system of the car. In addition, if the external POI identification module result is ambiguous (ex. more than one potential POI are identified), or the result of the feasibility analysis is negative, the system will try to solve these conflicts in collaboration with the user, through a dialog feedback system using the user interface of the vehicle. This feedback is also explored in Chapter VI related to **Research Question 2**.

Although it is not the current scope of this research, the system has a user identification module that analyzes if the user giving the command has the permissions to do it. This could be done by means of biometrics sensors or wearables; similar to the work described in [117].

In order to demonstrate the main functions of this system, this dissertation implements a simulation framework that is presented in the next subsection.

D. Simulation Framework

The main modules of the architecture presented in Figure 7 are simulated using a simulation framework composed by two components that interface with each other: a 3D virtual simulation environment and the core system processing environment.

1. 3D virtual environment

For the purpose of this research, the 3D virtual environment is composed by a vehicle model based on a simplified Ackermann car model [118] which position can be obtained (simulating a GPS sensor). For this research, the external sensors module is composed by a vision sensor with depth filter (being equivalent to the functioning of a stereo camera module). This vision sensor has a 70°x70° field of view and 256x256 pixels with a depth range of 2m to 65m. In addition, a ‘city style’ environment was set up. Making use of the available path planning module based on OMPL (Open Motion Planning Library) [119], which provides many sampling-based motion planning algorithms, the autonomous vehicle can navigate through this environment.

The data generated by this 3D environment, such as sensor data, vehicle position and the generated path points, is transmitted to the processing side for performing the main functions of our system.

2. System processing environment

The processing side of the presented simulator framework is in charge of the core operation of the system developed in this dissertation. In particular, it creates the user

intended command by means of the command constructor module, which uses the outputs of the speech and gesture recognition engines, and external POI identification module. The relevant information of this command (POI coordinates and task semantics in this dissertation) is transmitted to the 3D virtual environment side of the simulation framework, so its path planning module can update the pre-defined autonomous path. In addition, the feasibility analysis and voice based dialog feedback modules are also developed in this side of the framework.

In order to follow the vehicle's path, a path following module calculates the needed steering angles to follow the path calculated by the path planning module. These steering angles are transmitted to the vehicle model and controller of the 3D virtual environment side.

For the work presented in this dissertation, VREP [120] was used as the 3D virtual environment and MATLAB as the processing side. In addition, the for the speech recognition engine and dialog feedback system, the namespaces of *System.Speech* [121] were used. The functions of this library allows to perform speech recognition and synthesis processes needed for the speech recognition engine and dialog feedback module. In this way, as it will be explained in Chapter VI, following the *W3C Recommendation "Speech Recognition Grammar Specification"* [122], custom built grammars were developed in order to recognize the semantics of the user voice commands. Also, for some of the system functions, two open Robotics and Machine Vision toolboxes available in [123] were used.

Figure 8 shows the modules developed in each simulation side, as well as the main information exchanged between both. As it is show, the user performs a ‘simulated gesture’ by clicking on the vision sensor stream available in the processing environment. In order to make the simulated pointing gesture more realistic, some Gaussian bivariate noise was added to it. The pointing gesture makes possible to calculate the target POI needed for the system to function as it was explained in the previous subsection of this chapter and will be described further in this dissertation. In addition, for simplicity, this simulation framework lacks of more complex situations that a real life scenario has, such as pedestrian crossing, mixed traffic, or others. It is assumed that the car is able to handle those dynamics.

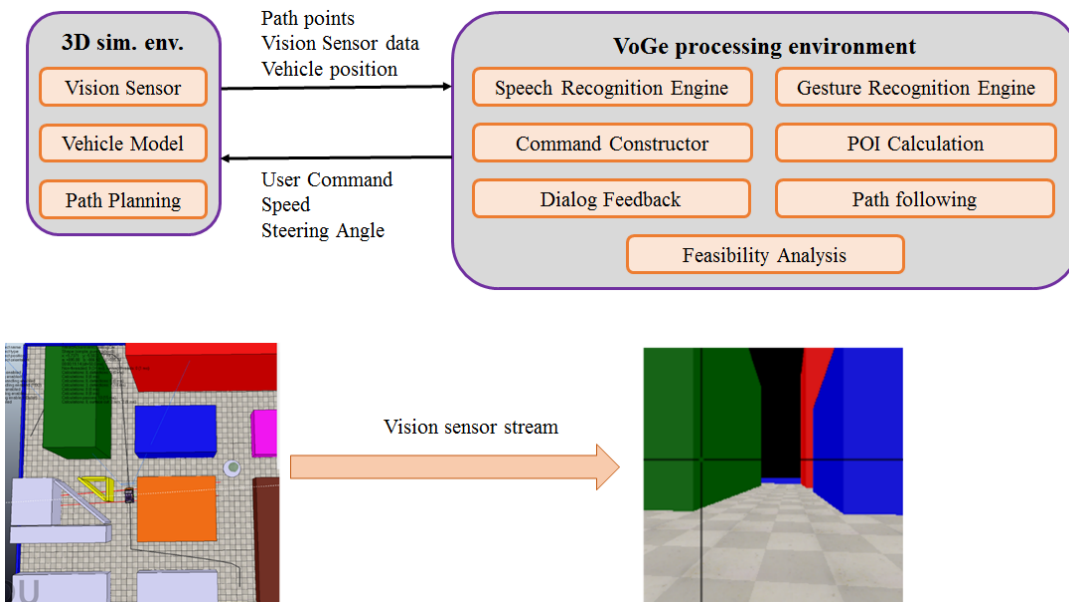


Figure 8. Simulation framework set up.

V. CHAPTER FIVE: POINT OF INTEREST COMPUTATION

A. Introduction

As it was stated in Chapter IV, one of the key elements of the command structure of the proposed system in this dissertation is based on the computation of the target POI that the user is pointing to. Thus, an external POI identification module is needed (Figure 7).

Regarding this, in Chapter IV it was mentioned that the vehicle external information was used to calculate the coordinates of the target POI. However, the range those sensors may be limited. A user may be pointing to a target location that is outside the sensor range. For this reason, the POI computation without the use of external sensors information must be also studied.

The literature review of Chapter III showed that for the particular case of situated awareness interactions [43]-[45], [88], the limited research uses some form of ray-casting techniques for the POI identification. This identification is in most of the cases based on the intersection of the pointing-ray with the location of the target building stored in a map database. In this way, the first POI that intersects the pointing ray is considered to be the target POI. This approach fails in cluttered environments, such as cities, where more than one POI could intersect the pointing ray. For example, a user may be pointing to a taller building behind the building that theoretically would be selected with the traditional ray-casting approaches. Figure 9 shows an example that illustrates this. The pointing rays intersect Building 1 and Building 2. The traditional approach would select Building 1 as the target POI; however, the target POI was Building 2.

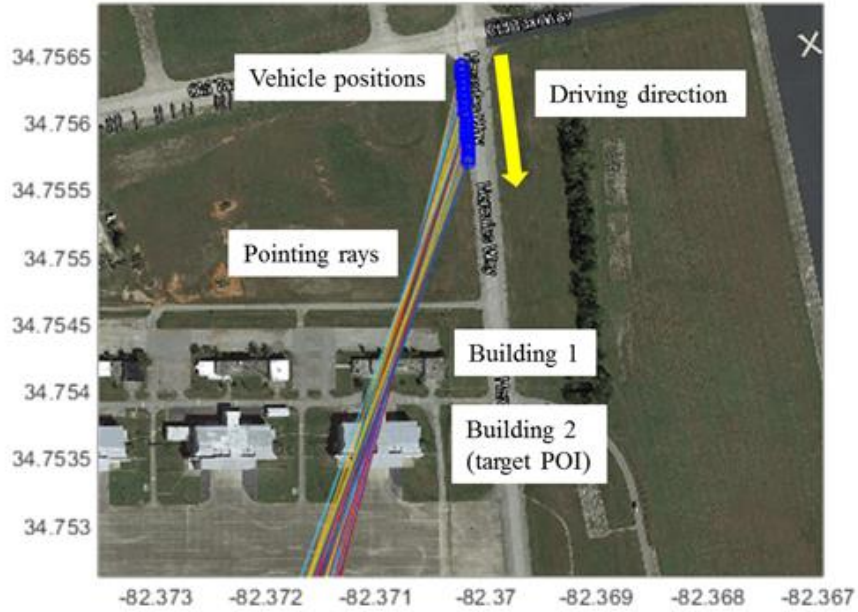


Figure 9. Dense scenario where traditional ray-casting fails.

The challenge of pointing in dense and cluttered environments has been identified as one of the main challenges of ray-casting techniques [89]-[91]. The use of pointing vector intersection has been proposed in [42], [90]; however, this limited research has not been applied on a moving vehicle use case. While the use of voice recognition has also been proposed to filter potential POIs [44], [45], there is still room for improvement in the automotive field.

This chapter elaborates on the POI computation challenge. In particular, it answers the **Research Question 1**: *How can the POI computation be improved for situated awareness interactions, especially in cluttered environments?*

The first part of this chapter presents a method based on pointing rays intersection to solve the POI computation issue, assuming that the POI is out of the vehicle sensors range.

In the second part, similar to the research done in the field of mobile manipulators in [124], vehicle sensor information to support the POI computation is included.

B. POI computation without using external sensor information

This part of the chapter develops a method based on pointing rays intersections to calculate the coordinates of the target POI when it is out of the range of the vehicle sensors.

1. Method

a. POI coordinates computation

The POI computation method presented in this part of the dissertation is based on the intersection of the pointing rays over the time. It is assumed that the vehicle has some type of Heads Up Display (HUD) technology available in the windshield of the car, and the HUD will show some visual feedback in the form of a pointer when it detects the pointing event (an example of this can be shown in Figure 1 of Chapter I). In this way, the user will tend to follow the pointer, changing the pointing angle over the time. As the car is moving, this will allow the pointing rays to intersect. This intersection will form the POI coordinates.

Figure 10 illustrates this concept.

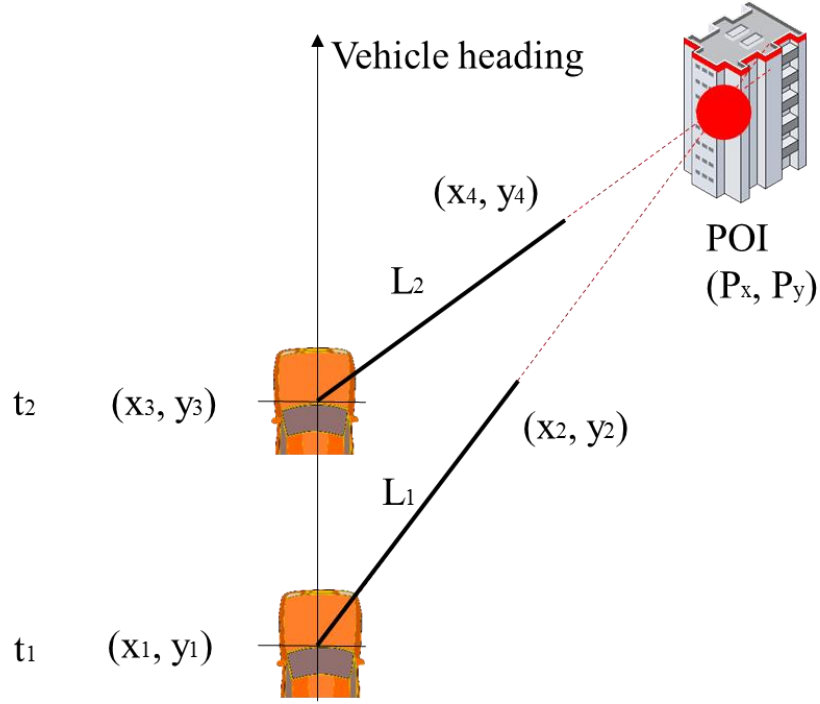


Figure 10. POI computation method based on pointing ray intersection.

Given two non-parallel pointing rays L_1 and L_2 with different start and end points at two different times t_1 and t_2 of a pointing sequence:

$$t_1 : L_1 \text{ defined by } (x_1, y_1) \text{ and } (x_2, y_2)$$

$$t_2 : L_2 \text{ defined by } (x_3, y_3) \text{ and } (x_4, y_4)$$

The intersection of the two pointing rays can be defined as [125]:

$$P_x = \frac{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \\ x_4 & 1 \end{vmatrix}}; P_y = \frac{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \end{vmatrix}}{\begin{vmatrix} y_1 & 1 \\ y_2 & 1 \\ y_3 & 1 \\ y_4 & 1 \end{vmatrix}} \quad (1)$$

Or:

$$(P_x, P_y) = \left(\frac{(x_1 y_2 - y_1 x_2)(x_3 - x_4) - (x_1 - x_2)(x_3 y_4 - y_3 x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}, \right. \\ \left. \frac{(x_1 y_2 - y_1 x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 y_4 - y_3 x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \right) \quad (2)$$

Considering the pointing sequence formed by the set of pointing

vectors $\{L_i\}_{1 \leq i \leq N}$ for $N > 0$, and assuming that none of them are parallel, $J = \frac{N \cdot (N-1)}{2}$

intersection points can be calculated applying equations (1) or (2). The result is the set of

intersection points $\{(P_x, P_y)_k\}_{1 \leq k \leq J}$. The average of the intersection points of this set

defines the coordinates of the target POI.

$$POI \Rightarrow (x_{POI}, y_{POI}) = \text{avg} \left(\{(P_x, P_y)_k\}_{1 \leq k \leq J} \right) \quad (3)$$

This method is expected to improve the computation of the target POI when compared to the traditional ray-casting method. In special, under dense and cluttered environments.

b. POI identification

Once the coordinates of the POI are calculated, the next step is to compare them to the coordinates of the map elements available in an annotated map. For the purpose of this research this map database has the following fields for each building.

Map element	
Field	Data type
Name	string
Polygon Coordinates	Array _{Nx2} <lon, lat>

Given the coordinates of the computed POI (x_{POI} , y_{POI}), and the matrix of the distances to the closest polygon coordinate of each of the M map elements

$D_{1 \times M} = [d_i]_{1 \leq i \leq M}$. The selected POI is the one with $\min(d_j)$ (i.e. the one which polygon coordinate is closer to the calculated POI coordinates).

As it will be mentioned in Chapter VII, it can happen that more than one POI is selected using this method (i.e. the distances are similar). In this case, the dialog feedback system described in Chapter VI can help to solve this ambiguous situation.

Table 2 shows a result example. In this case, as it can also be seen in Figure 11, the red building (blue asterisk) was selected as the target POI (minimum distance to the calculated POI coordinates).

Table 2. POI selection example.

Distance	Building Name
1.5523	red
5.5533	blue
14.8985	green
16.3972	pink
18.7790	orange
23.6936	yellow
24.4399	brown
31.3969	lightblue
31.7174	white
37.7900	pink
40.9031	grey

- Path
- - - Followed Path
- Start position
- End position
- * Computed POI: Intersection method
- ★ Computed POI: sensor method
- ◆ POI ground of truth

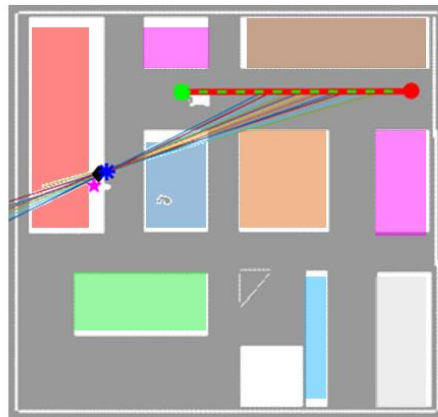


Figure 11. POI selection example map.

c. Pointing vectors calculation

The proposed method is based on the intersection of the pointing rays (vectors).

Thus, it is needed to explain how those rays are calculated.

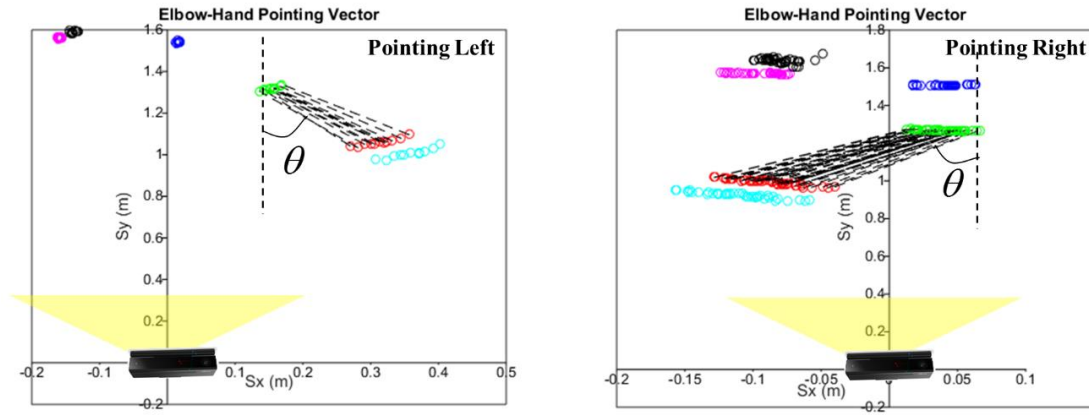


Figure 12. Pointing vectors example.

Given a pointing gesture performed by a user, it is possible to determine its corresponding sequence of pointing vectors. An example of a top view of these vectors in a pointing sequence is shown in Figure 12. For this particular case, as it will be explained later in this chapter, the elbow and hand joints positions were used to form these pointing vectors.

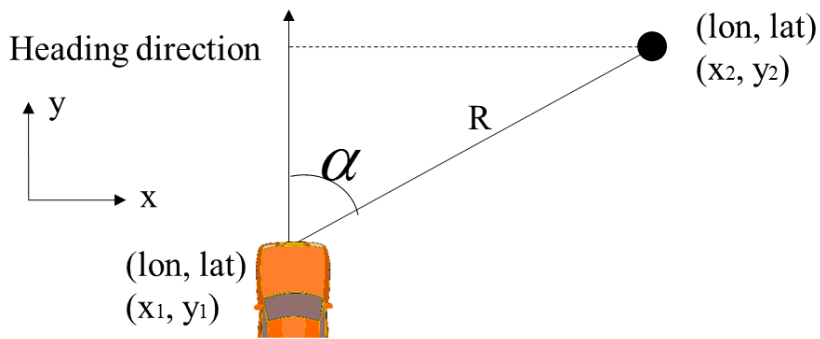


Figure 13. Pointing ray calculation for each vehicle position.

Using the commonly known equation of a line, it is possible to calculate the pointing angle θ for each pointing vector inside the car. Adding this pointing angle to the heading angle of the car, the pointing ray for each vehicle position can be calculated following the method depicted in Figure 13.

With,

$$\begin{aligned} x_2 &= x_1 + R.\sin(\alpha) \\ y_2 &= y_1 + R.\cos(\alpha) \end{aligned} \tag{4}$$

Where,

$$\begin{aligned} \alpha(^{\circ}) &= car_heading + \theta \\ \text{and } R &> 0 \end{aligned} \tag{5}$$

Thus, for a pointing sequence, a set of pointing rays can be formed, which intersection, as it was explained, can be used to determine the coordinates of the target POI.

2. Simulation Results

To validate the proposed method, a simulated a pointing scenario using the simulation framework described in Chapter IV was developed. As it was explained, the pointing gesture is simulated by a ‘mouse click’ (see Figure 8) (equation (9) further described in this chapter shows the simulated pointing angle calculation). We added some Gaussian bivariate noise to this pointing gesture in order to have more realistic results.

The proposed scenario assumes that the vehicle has the characteristics described in Chapter I. In this case, the role of the vision sensor is only to serve as the simulated

HUD (windshield of the car) (i.e. no depth information is available). In addition, for simplicity, this scenario lacks of more complex situations that a real life scenario has, such as pedestrian crossing, mixed traffic, or others. It is assumed that the car is able to handle those dynamics. Moreover, the pointing is performed when the car is driving in straight line.

For performing these simulations a ground of truth POI was inserted into the simulation scenario. In this way, the distance error between that POI and the resulting one from our method can be compared.

A total of 383 simulations were performed: 166 for non-cluttered conditions (i.e., pointing to the first building in the field of view without any building in front), and 217 for cluttered conditions (i.e. pointing to a building behind the first building in the field of view)

The simulations performed resulted in a distance error in non-cluttered conditions of $2.67 \pm 2.45\text{m}$ with respect the ground of truth. For cluttered conditions this distance error increased to $4.10 \pm 2.90\text{m}$.

In order to compare our POI computation method with the traditional one based on ray-casting, the accuracy (in %) of them was compared. In this way, a computed POI is considered accurate if the proposed method in this dissertation selects the correct target building. In a similar way, a pointing ray is considered to be accurate if it intersects the target building and no ambiguity is detected (i.e. the pointing ray does not intersect other buildings).

Table 3 shows the accuracy results from the executed simulations. As it can be seen the method proposed in this dissertation performs better than the traditional ray-casting one. This is especially true in dense environments, where several buildings may intersect the pointing rays.

Table 3. POI identification. Simulation accuracy results.

POI computation method		Intersection based	Ray-casting based
Accuracy (%)	Non-cluttered	96%	100%
	Cluttered	76.5%	0%
	Total	86.25%	50%

- Path
- - - Followed Path
- Start position
- End position
- ✱ Computed POI: Intersection method
- ★ Computed POI: sensor method
- ◆ POI ground of truth

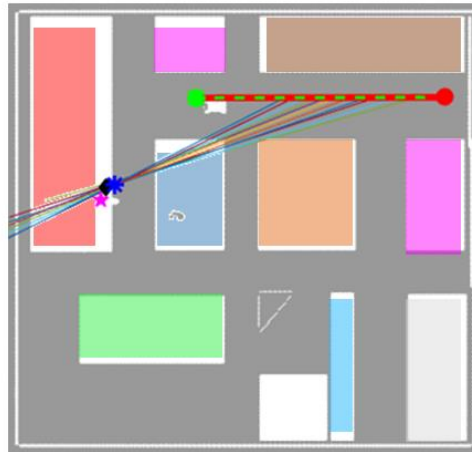


Figure 14. POI computation methods comparison.

Figure 14 illustrates this concept. The method based on ray-casting would result in two potential target buildings (blue and red buildings), generating POI computation

ambiguity. However, the intersection based method provides a non-ambiguous and correct result (red building).

However, as it was seen in Table 3, the method developed in this dissertation is 13.75% inaccurate in total (4% for non-cluttered and 23.5% for cluttered scenarios). For this reason, as it will be described in Chapter VII an additional disambiguation step must be added to solve them.

3. Real world data collection results

The objective of this part of the dissertation is to compare the simulation results with real world ones, and identify the challenges that a potential real world implementation may have.

For this, an applications was developed for collecting pointing gestures and GPS data.

a. Apparatus

For the implementation of this application two main hardware components where used:

1) Microsoft Kinect Sensor v2 (Figure 15) [126]

Developed by Microsoft, the Microsoft Kinect Sensor v2 counts with a color (RGB) camera and a Time of Flight (ToF) camera that allows to get body joints information. In particular, it can recognize 25 body joints (Figure 15). It also has an array of microphones to capture sound. The Microsoft Kinect SDK has tools and APIs that

makes relatively easy the development of body tracking, face tracking and voice recognition applications.

The Microsoft Kinect Sensor v2 was used in this research as the Interior Sensors module of the system architecture depicted in Figure 7.

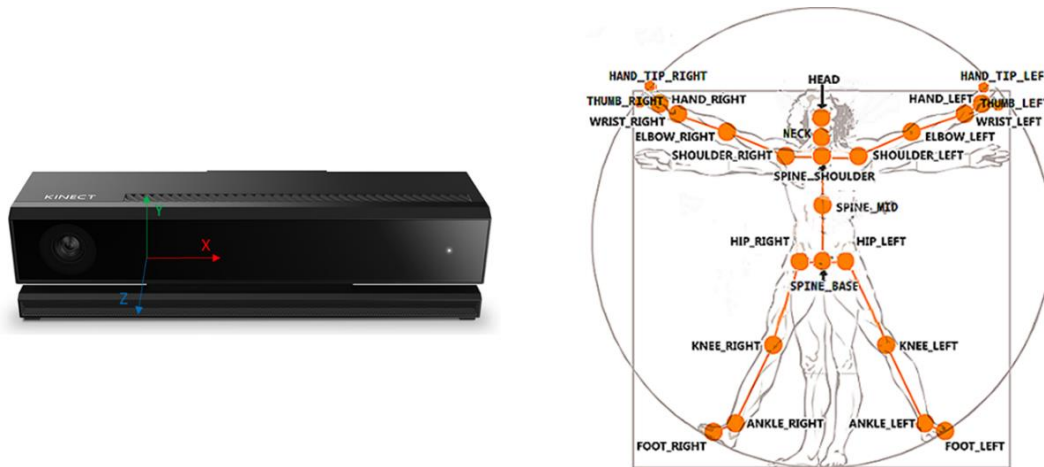


Figure 15. Microsoft Kinect Sensor v2 and Tracked Body Joints [126].

2) Globalsat ND-105C Micro USB GPS Receiver (Figure 16) [127]

This GPS receiver has a micro USB interface, so it can be connected to a computer to receive NMEA sentences [128] at an update rate of 1 Hz.



Figure 16. GPS receiver.

b. Development of a Pointing Gestures data collection application.

The pointing gestures application combines the tracked body data and the GPS receiver data into a single txt file. Its block diagram is depicted in Figure 17.

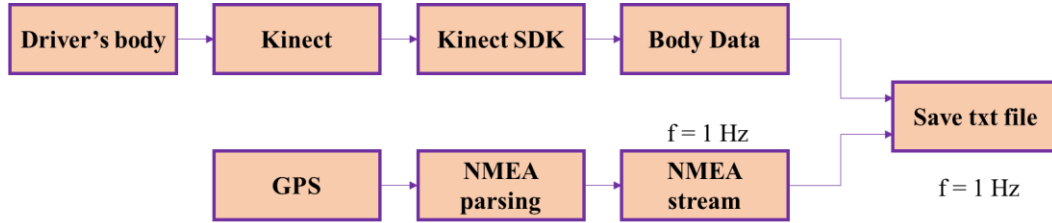


Figure 17. Pointing Application building blocks.

The Microsoft Kinect tracks the position of the body joints of the users via Kinect API functions [126]. For the purpose of this research, as the only interest is in pointing gestures, the tracked body parts positions (in meters) are: *left hand, left elbow, left shoulder, center shoulder* and *head*. The right part of the body could also have been tracked; however, for simplicity only the left side was tracked. This is not considered a critical aspect of the application due to the fact that only minor software updates would be required.

The GPS is receiving data at a rate of 1 Hz. A parsing phase is needed in order to convert NMEA raw traces into readable ones. This parsed NMEA stream is combined with the user's tracked body joints data in the same line, which is saved in a txt file every second. The structure of the blocks of data that are saved in this file is shown in Figure 18.

GPS_time	Latitude	Longitude	Speed (kmh)	Heading Direction (degrees)	X_left_hand	Y_left_hand	Z_left_hand
X_left_hand_tip	Y_left_hand_tip	Z_left_hand_tip	X_left_elbow	Y_left_elbow	Z_left_elbow	X_left_shoulder	Y_left_shoulder
Z_left_shoulder	X_center_shoulder	Y_center_shoulder	Z_center_shoulder	X_head	Y_head	Z_head	*Newline

From GPS
 From Kinect

1 sample (row) per second (based on GPS update frequency of 1 Hz)

*Joint position in meters

Figure 18. File structure for the pointing application.

Figure 19 shows the basic user interface created for this application. It is important to note that the objective of this application is the data collection and not the user use. Thus, efforts were focused in other areas and not in the development of a friendly user interface. The tracking application is based on the Microsoft Kinect application examples available in [129], [130]. The NMEA parser is based on the example available in [131].

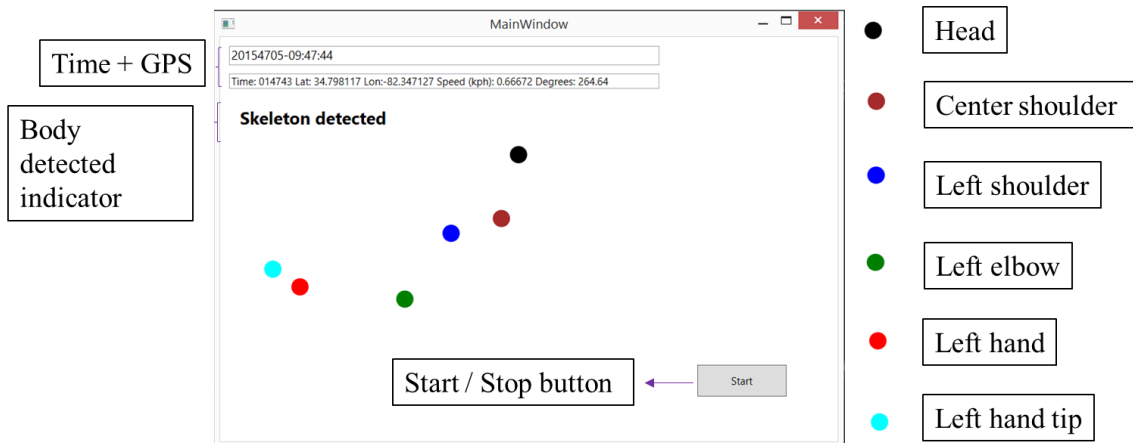


Figure 19. User interface of the pointing application.

c. Data Collection Scenarios

The data collection was done in the International Transportation Innovation Center (ITIC) test track facilities. In particular, two main sections of the test track were used:

- 1) **Section 1** (Figure 20 – Scenario 1) had only one target POI (red dot in Figure 20). In addition two driving directions (yellow arrow) tests were done, allowing pointing to the left or right.
- 2) **Section 2** (Figure 20 – Scenario 2) constitutes the most interesting case for these tests. There are two potential POI to be selected when pointing right in the driving direction (yellow arrow). Building 2 is taller than Building 1. In this case, the target POI was Building 2 (red dot in Figure 20). As it was already stated, this scenario is important to analyze the pointing strategy presented in this research in dense environments, where the traditional pointing strategies fail.

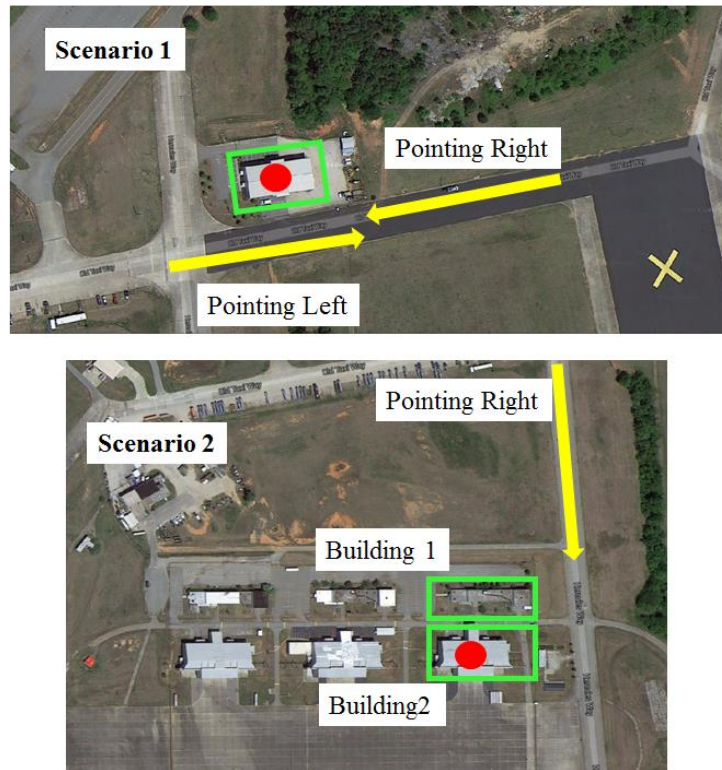


Figure 20. Satellite view of the data collection scenarios.

The hardware described previously was set up in a vehicle. In this way, the Kinect sensor as well as the GPS receiver were connected to a Laptop Computer that executed the developed pointing application. In addition a GoPro camera was also mounted; however its use is not relevant for the data collection. Figure 21 shows the experiment set up.

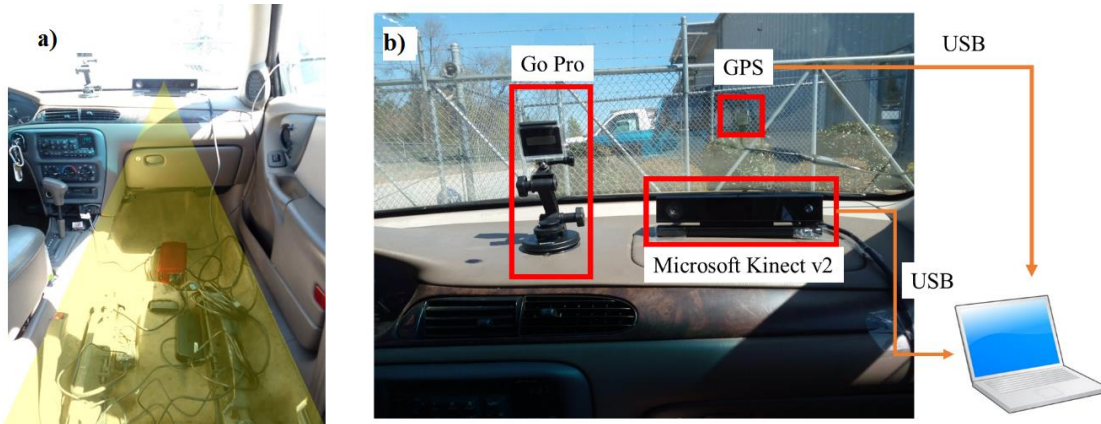


Figure 21. a) Experiment set up. b) Detailed view.

The Microsoft Kinect sensor needs a minimum distance of 50 cms. for tracking the human body. For this reason, as it can be seen in Figure 21, the passenger seat of the experimental vehicle was removed and the experimenter performed the pointing gestures from the rear seat of the car. In addition, for these experiments the vehicle was always in ‘Drive’ without pressing the acceleration or braking pedal. This was done to try to maintain a constant speed during the data collection process (~5 mph).

d. Procedure

For the *pointing application*, the experimenter pointed with his left hand to the target buildings of Scenario 1 and 2 of Figure 20. As it was already described, it was assumed that in the future a HUD based windshield will give some visual feedback of a pointer generated by the pointing gesture. Thus, the hand of the experimenter was

tracking the target POI as the vehicle was moving, causing a change in the pointing angle.

Due to challenges in the functioning of the used instrumentation hardware of the experiment set up, a total of 13 valid data collection experiments were performed by the author of this research: 7 of them for non-cluttered environments, and 6 for cluttered environments.

e. Data Analysis

The data collected from the pointing gesture sequences was analyzed following these steps:

1) Coordinates transformation to standard coordinate system

The Microsoft Kinect coordinate system is different from the standard one. Thus, a transformation of the body parts coordinates is required. Figure 22 shows this transformation [132].

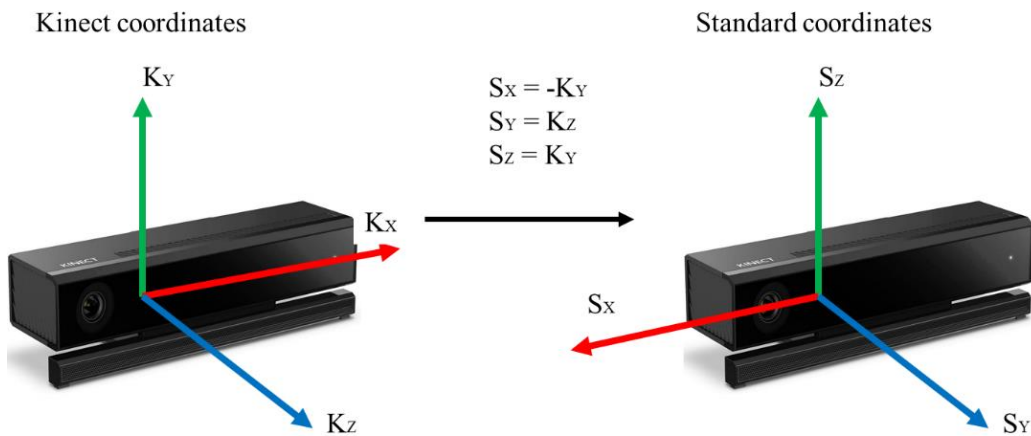


Figure 22. Coordinates transformation [132].

2) Pointing samples identification

During the data collection process there were samples that corresponded to the same GPS coordinates (samples captured at the beginning and the end of the data collection process, when the vehicle was stopped). The average of the samples with the same GPS position was done in order to avoid having body joint positions in the same GPS coordinate. That would have caused to have different pointing vectors for the same vehicle position.

In addition, even when the vehicle was moving there could be samples not corresponding to the pointing part of the pointing sequence. Research has been done to differentiate the pointing and non-pointing parts of a pointing sequence, mainly using Hidden Markov Models (HMM) [95], [133]-[135]. However, the objective of this research is not to develop methods for differentiating pointing sequence parts. In addition, more training data would have been needed. For this reason, in order to filter the pointing and no-pointing parts of each pointing sequence, the c-means algorithm (with $c=2$) was used. c-means is an unsupervised learning technique that allows classifying unlabeled data in 'c' different classes [136].

For this case, the classes are 'Pointing' and 'No pointing'. Thus, $c=2$. In addition, the assumption is that a user will spend more time during the 'pointing' part of the sequence than during the 'no pointing' part. In this way, the class with more number of feature vectors is assumed to be the 'Pointing' class.

The c-means algorithm was used twice. The first time the hand and elbow positions (x , y , z – in meters) were used to form the feature vectors. This was done because, as it will be explained, the pointing vector was chosen to be the one formed by the hand and elbow joints' positions. Using empirical tests, it was decided to use the hand joint 'z coordinate' for second application of c-means. This last application of c-means filtered some spurious samples of the pointing sequence.

Figure 23 shows an example of the results of this process.

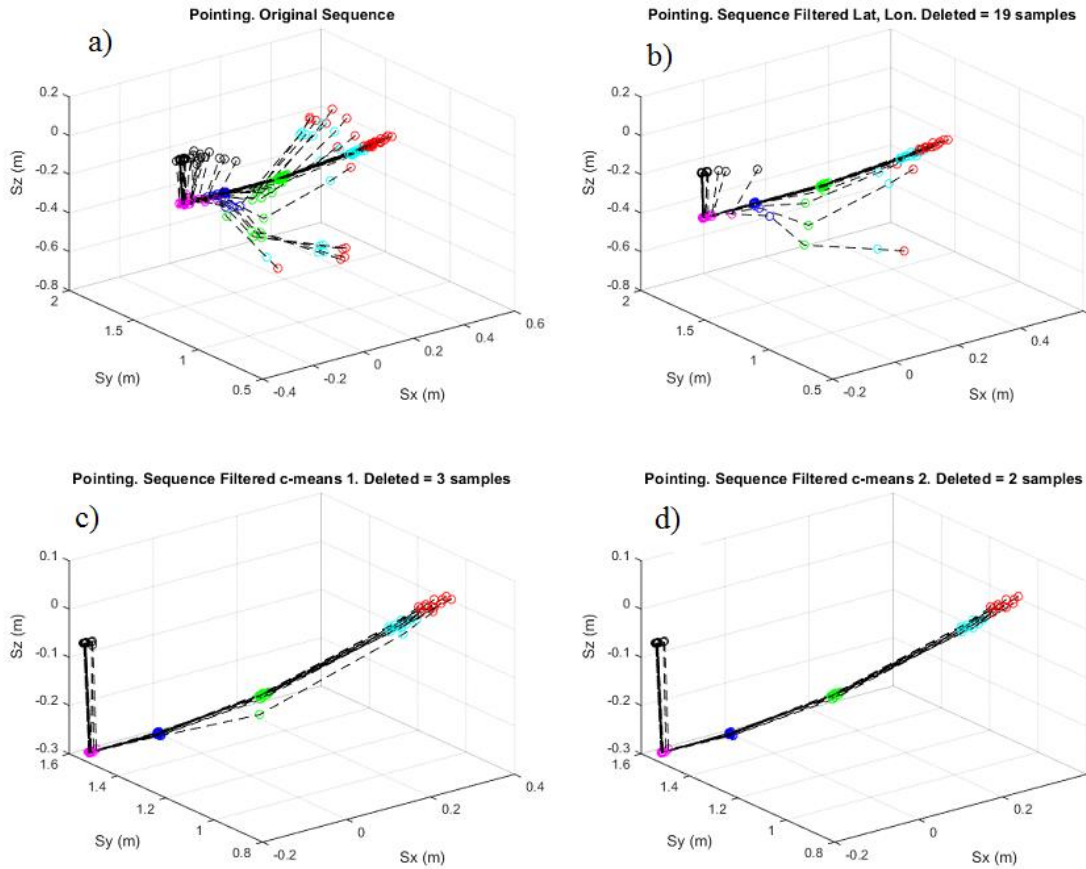


Figure 23. a) Original sequence. b) Same coordinates filtered. c) c-means loop 1. d) c-means loop 2.

As it is show in Figure 23, this method filters well the pointing parts of the pointing sequence. However, it needs to be recognized that for a real time implementation, HMM is desired. Thus, training data may be needed. However, this is not the focus of this research.

Figure 24 shows the position of the Kinect Sensor in the plot and the meaning of each data point.

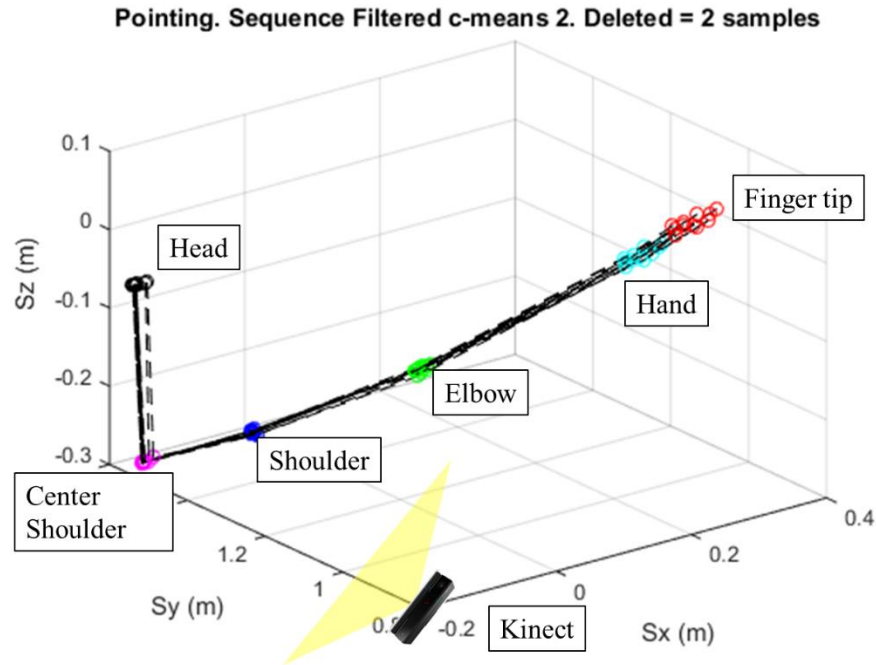


Figure 24. Body parts joints positions meaning.

3) Pointing vector calculation

Once the pointing data is filtered, the pointing vector has to be calculated following the method depicted in Figure 13 of this chapter.

Figure 25 shows an example of pointing rays using the collected data.

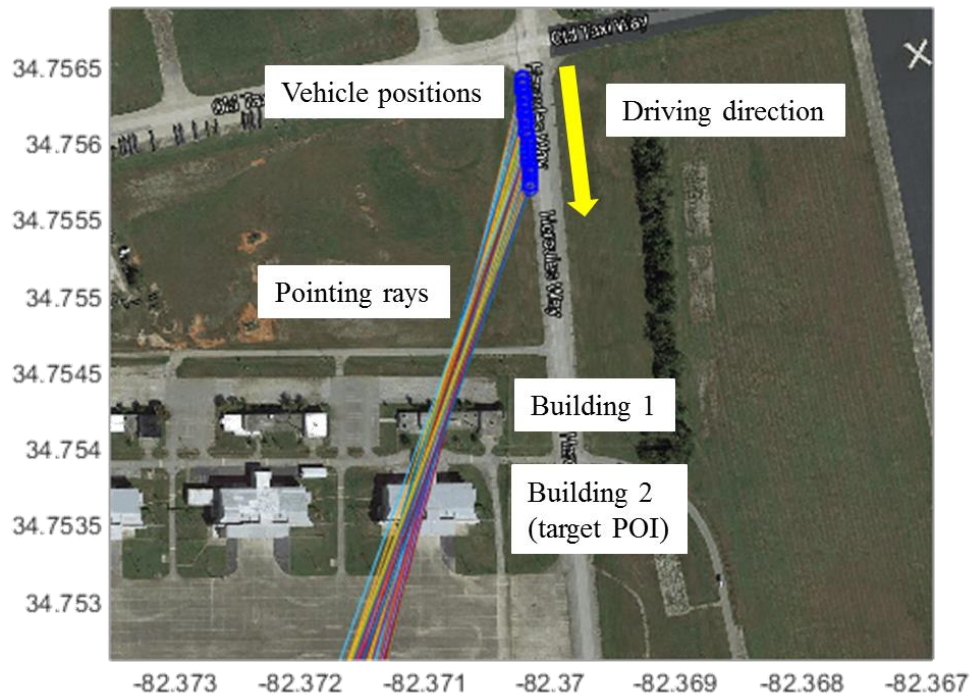


Figure 25. Pointing rays for each vehicle position (pointing right case).

As it can be seen in Figure 25, the pointing rays for each vehicle position cross Buildings 1 and 2. As it was already mentioned, for the traditional ray casting approaches, Building 1 would be the result of the POI calculation. However, the real target POI in this experiment was Building 2. For this reason, the approach based on pointing rays' intersections proposed in this research is needed. This will be the next step of the process.

4) Point of Interest calculation based on pointing rays' intersections

The calculation of the pointing rays included the computation of two points for each ray (equation (4)). Thus, the POI calculation based on pointing rays intersection

(equations (1) or (2), and (3)) can be used, and the GPS coordinates of the target POI determined.

Figure 26 shows in red the intersection points and in a blue asterisk the average of them; that is, the calculated coordinates of target POI. As it can be seen, the resulting POI coordinates are the correct ones (Building 2).

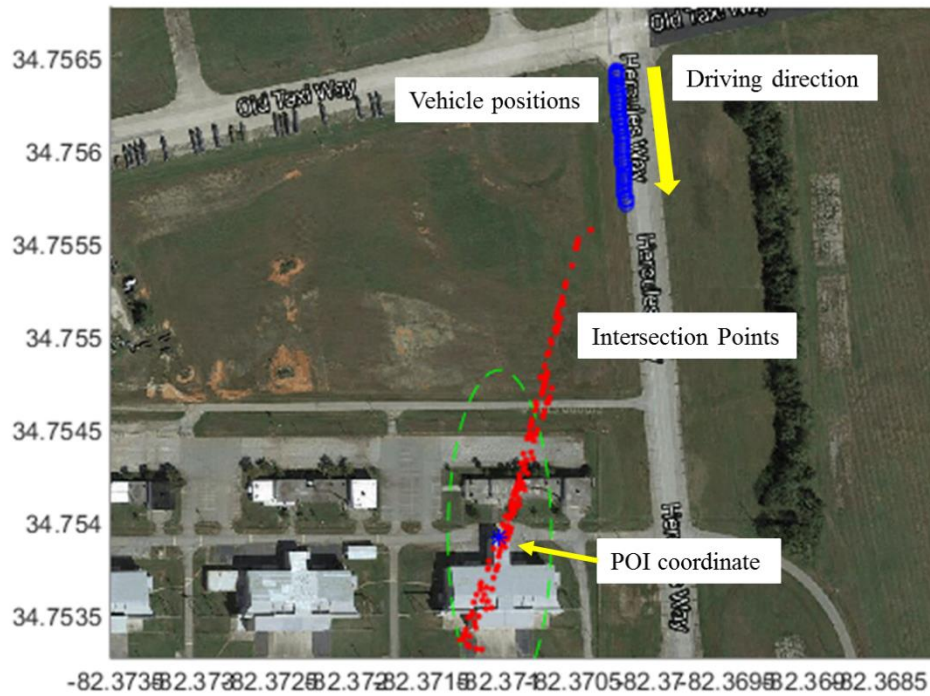


Figure 26. POI calculation as a result of pointing rays intersections.

The green dotted ellipse of Figure 26 is the standard deviation of the intersection points. In order to reduce the variability of them, an Extended Kalman Filter (EKF) [137], [138] was implemented.

5) Extended Kalman Filter (EKF)

For this research, the states of interest are the ones of the target POI. That is, the states of the intersection points calculated in the previous step. As it is shown in Figure 27 the POI can be modeled as a dynamic object with respect to the vehicle's coordinate system. This can be demonstrated by doing vector operations of equation (6):

$$\vec{r}_{tw} = \vec{r}_{cw} + \vec{r}_{tc} \rightarrow \vec{v}_{tw} = \vec{v}_{cw} + \vec{v}_{tc} \xrightarrow{\vec{v}_{tw}=0} \vec{v}_{tc} = -\vec{v}_{cw} \quad (6)$$

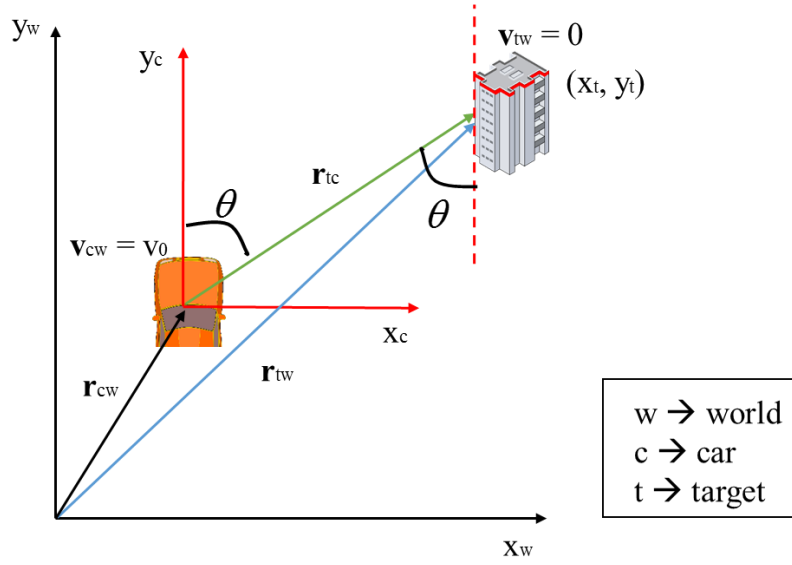


Figure 27. POI modeled as a dynamic object with respect to the coordinate system of the car.

Defining the state vector as the one formed by the target POI positions and velocities, $X = (x \ y \ \dot{x} \ \dot{y})^T$; the computations for the EKF are the following:

1) Initialization step, formed by:

- Initial states vector $X_0 = (P_{x_0} \ P_{y_0} \ 0 \ 0)^T$; being P_{x_0} and P_{y_0} the coordinates of an initial intersection point calculated in the previous step.
- Initial uncertainty matrix $P_0 = \begin{pmatrix} 4.I_{2 \times 2} & 0_{2 \times 2} \\ 0_{2 \times 2} & 10^{-5}.I_{2 \times 2} \end{pmatrix}$

2) Measurement step.

$$\begin{aligned} K_k &= P_k^- H_k^T (H_k P_k^- H_k^T + R)^{-1} \\ \hat{X}_k &= \hat{X}_k^- + K_k (Y_k - h(\hat{X}_k^-)) \\ P_k &= (I - K_k H_k) P_k^- \end{aligned} \tag{7}$$

3) Prediction step.

$$\begin{aligned} \hat{X}_{k+1}^- &= F \hat{X}_k \\ P_{k+1}^- &= F P_k F^T + Q \end{aligned} \tag{8}$$

Where,

- State vector: $X = (x \ y \ \dot{x} \ \dot{y})^T$
- Transition matrix (assuming constant velocities): $F = \begin{pmatrix} I_{2 \times 2} & dt.I_{2 \times 2} \\ 0_{2 \times 2} & I_{2 \times 2} \end{pmatrix}$

With time-step $dt = 1$.

- Uncertainty matrix P
- Covariance matrix: $Q = 0.1.I_{4 \times 4}$

- Measurement vector formed by the intersection coordinates and its corresponding pointing angle: $Y_k = (x_k \quad y_k \quad \theta_k)^T$
- Measurement nonlinear function: $h(X_k) = \left(x_k \quad y_k \quad \arctan\left(\frac{x_k}{y_k}\right) \right)^T$
- The function h is linearized applying its Jacobian. This replaces the measurement matrix; so: $H_k = \begin{pmatrix} I_{2 \times 2} & 0_{2 \times 1} \\ A_{1 \times 2} & 0_{1 \times 1} \end{pmatrix}$

Where,

$$A = \begin{pmatrix} \frac{1/y_k^-}{1 + \left(x_k^-/y_k^-\right)^2} & -\frac{x_k^-/y_k^-}{1 + \left(x_k^-/y_k^-\right)^2} \end{pmatrix}$$

- Measurement noise: $R = \begin{pmatrix} 4 \cdot I_{2 \times 2} & 0_{2 \times 1} \\ 0_{1 \times 2} & 10^{-5} \end{pmatrix}$
- Kalman gain K .
- Identity matrix $I = I_{4 \times 4}$

The application of this process allows to reduce the measurements (intersection points) variability. The average of these measurements (that is, the resulting target POI coordinates) do not seem to be affected by the EKF for all the cases. Figure 28 shows the results of the EKF in pink dots. For this particular case, the measurement standard deviation is nearly affected. However, as it is shown in Figure 29, the standard deviation after the EKF application (blue dotted ellipse) is smaller than before applying it (green dotted ellipse).

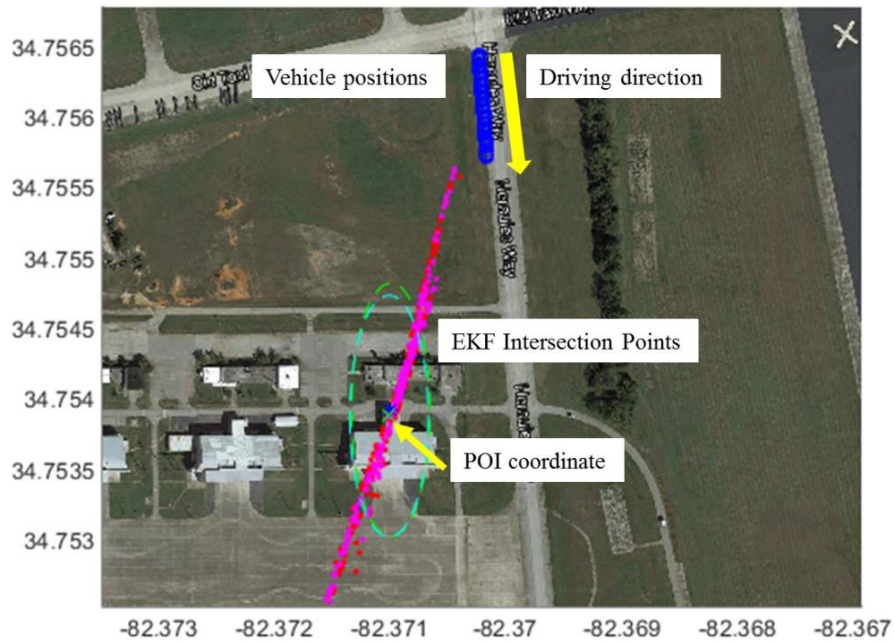


Figure 28. EKF results.

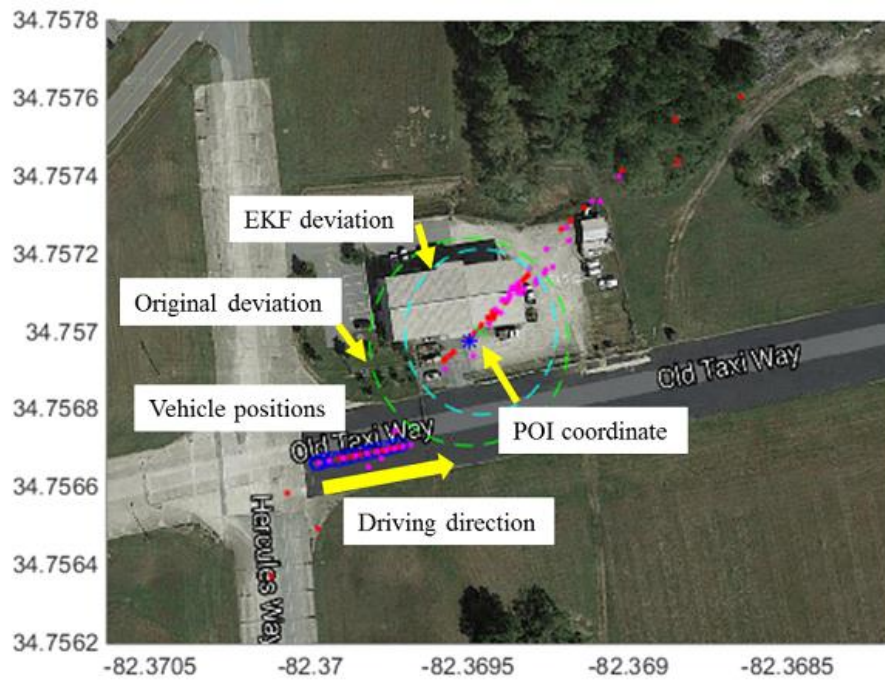


Figure 29. EKF results. Change in measurements deviations.

f. Results

Table 4 shows the POI identification results in terms of accuracy rate.

Table 4. POI identification. Real world accuracy results.

POI computation method		Intersection based	Ray-casting based
Accuracy (%)	Non-cluttered	100%	100%
	Cluttered	33.3%	0%
	Total	66.65%	50%

As it can be seen, for the experiments performed the traditional ray-casting. A computed POI is considered accurate if the proposed method in this dissertation selects the correct target building. In a similar way, a pointing ray is considered to be accurate if it intersects the target building and no ambiguity is detected (i.e. the pointing ray does not intersect other buildings). This method and the one proposed in this dissertation have the same accuracy rates for non-cluttered environments, as in that scenario only one building is in the map database. The major difference between both methods is seen in cluttered environments. In this scenario, the method proposed in this dissertation outperforms by 33.3% the traditional ray-casting method. The pointing rays intersect the two potential target POIs, generating a POI computation ambiguity. This makes the method of this dissertation 16.65% more accurate in total.

However, as it can be seen in Figure 30 and Table 4, some ambiguities in the POI computation may exist. In this case, the calculated target POI is not accurate. Moreover, neither the intersection points, nor the pointing rays cross any of the potential

target buildings. For this reason, as it will be described in Chapter VII an additional disambiguation step must be added to solve them.

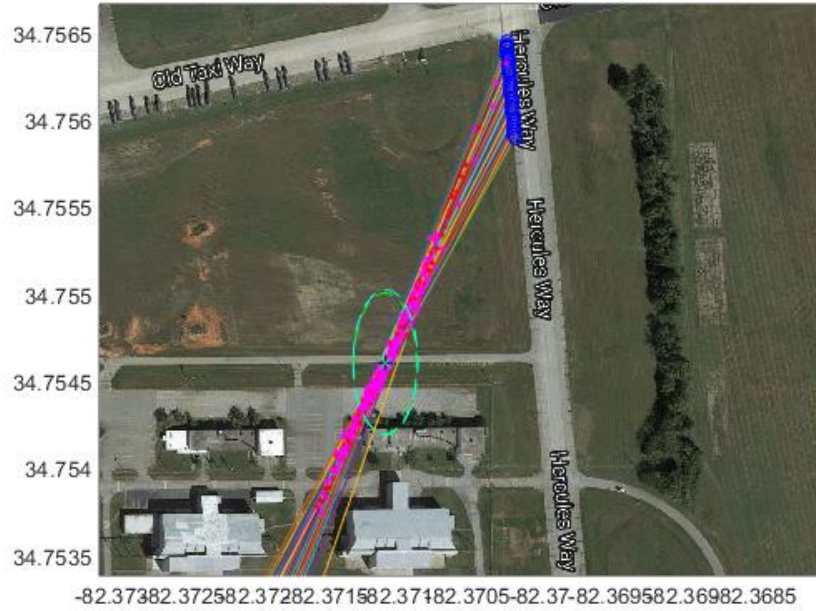


Figure 30. POI computation ambiguity.

In addition, it is important to mention the limited amount of successful data collected during the real world experiments. This was caused, as it will be explained in the next subsection, by limitations in the instrumentation hardware used for the experiment set up.

4. Comparison simulation and real world results

As it can be seen in Table 3 and Table 4 there is a considerable accuracy difference between the results of the simulations and the real world experiments. In addition, the number of successful data collection experiments is considerable low in the real world experiments when compared with the simulation ones.

During the real world experiments performed in this dissertation two main challenges were identified:

1) User feedback. In Chapter I it was assumed that the vehicle in which this research will be applied would be equipped with heads up display (HUD) technology in its windshield, so that the intersection of the pointing ray and the windshield could be highlighted, providing some feedback to the user regarding the target POI. The vehicle used did not have this technology available. Thus, this feedback that would have assisted to the pointing process was not produced. Taking into account that the pointing angle differences cause higher lateral errors at longer distances, this feedback would be essential to perform this real world experiments.

2) Hardware instrumentation functioning challenges. The ToF camera used for tracking the user pointing gesture [126] has functioning challenges when it is mounted on surfaces that are subject to vibration. As it was shown in Figure 21, this camera was mounted on the dashboard of the vehicle used for these experiments. The vibration produced by the vehicle engine as well as the road, made the camera stop tracking the user. This is the main reason of the low number of successful experiments performed.

As it was mentioned in Chapter I, it is assumed that the vehicle in which this research would be applied is perfectly instrumented, with automotive grade hardware components. Thus, potentially improving the real world results closer to simulation levels.

As it was described in Chapter IV, external sensors information can be used to compute the POI if it is in their range. Next subsection develops a method to perform this computation.

C. **POI computation with external sensor information**

In situations where the target POI is in the range of the external sensors of the autonomous vehicle (LIDAR, radar or others), the intersection between the pointing ray and the Point Cloud of the sensors can support in the POI computation process. Implementing a strategy to calculate the coordinates of the target POI based on this is the objective of this part of the dissertation.

1. **Method**

Using the simulation framework described in Chapter IV, this section develops a method to calculate the target POI coordinates when it is in the sensors range.

The data received from the vision sensor in the processing side of our simulator contains 3D information. When a user points to a pixel p with coordinates (C_x, C_y) of the 2D streamed image, its *depth* value is available and the coordinates of the POI can be

calculated using the following perspective projection method adapted to our simulation framework [139], [140]. The basics of this method can be depicted in Figure 31.

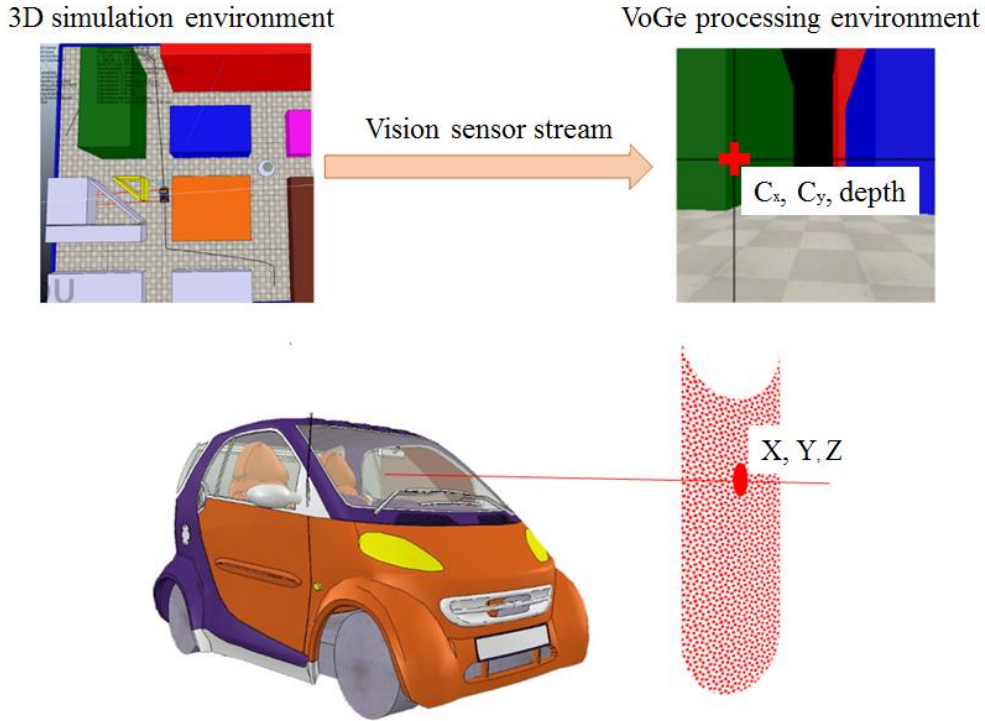


Figure 31. Illustration of POI computation with external sensor data.

Considering a vision sensor with depth filter and its parameters perspective angle γ in degrees, x and y resolutions (r_x, r_y) in pixels, the angles that form the pixel coordinates of the POI with respect to the vision sensor can be calculated using (9) (the equation of θ was used in the computation of POI without sensor information to simulate the pointing angle in the simulation environment).

$$\begin{aligned}
\theta[rad] &= \left[\frac{\left(\frac{r_x}{2} - C_x - \frac{1}{2} \right)}{\frac{r_x}{2}} \right] \cdot \alpha \rightarrow C_x \text{ angle} \\
\varphi[rad] &= \left[\frac{\left(C_y - \frac{r_y}{2} + \frac{1}{2} \right)}{\frac{r_y}{2}} \right] \cdot \beta \rightarrow C_y \text{ angle}
\end{aligned} \tag{9}$$

Where $\alpha = \frac{\gamma}{2} \cdot \frac{\pi}{180}$ and $\beta = \alpha \cdot \frac{r_y}{r_x}$ represent the half perspective angles with respect to x and y in radians.

The real world vector of coordinates of the pointed POI $W_{3 \times 1} = [x \ y \ z]$ can be calculated by $W_{3 \times 1} = M_{3 \times 4} \cdot P_{4 \times 1}$, where $M_{3 \times 4}$ is the transformation matrix of the vision sensor and $P_{4 \times 1} = [x' \ y' \ z' \ 1]^T$ the coordinates vector of the POI relative to the vision sensor, given by (10)

$$\begin{aligned}
z' &= Z_{near} + Z_{far} \cdot depth \\
x' &= \tan(\theta) \cdot z' \\
y' &= -\tan(\varphi) \cdot z'
\end{aligned} \tag{10}$$

being the parameters Z_{near} and Z_{far} the planes of the vision sensor field of view in *meters*.

2. Simulation Results

Following the same simulation process described already in this chapter, we performed the pointing simulations using external sensor information.

A total of 383 simulations were performed: 166 for non-cluttered conditions (i.e., pointing to the first building in the field of view without any building in front), and 217 for cluttered conditions (i.e. pointing to a building behind the first building in the field of view).

The simulations performed resulted in a distance error in non-cluttered conditions of $0.95 \pm 0.09\text{m}$ with respect the ground of truth. For cluttered conditions this distance error was $1.24 \pm 0.28\text{m}$.

As it was expected, the distance error is lower than the intersection method presented before because there is more information available (depth information). In addition, no errors were detected in the POI identification (i.e. the computed POI always hit the target). However, as we already saw in this chapter, a real world scenario is subject to more sources of noise and inaccuracies. For this reason, as it will be explained in Chapters VI and VII, a disambiguation step is needed.

Figure 14 also illustrates a result of the POI computation using external sensor information.

D. POI information retrieval

Once the coordinates of target POI are calculated, it is possible to retrieve relevant information about it, such as type of POI, opening hours, or other. As it was already explained in this chapter, the coordinates of the POI can be used to pull information from an annotated map.

However, as in Chapter I it was assumed that our car would have some HUD technology available, it could be possible to retrieve information directly from the projections of the HUD, for example, color information of the target POI.

By using the RGB information of the pointed pixel of the HUD and functions of the machine vision toolbox developed in [123], the color of the POI of our simulation environment can be added to the final structure of the user command. In this way, as it will be explained in Chapters VI and VII, the dialog feedback system can use this additional information to confirm or disambiguate the intended user command.

Table 5 shows an example of this command structure. As it can be seen, the color information is included in it.

Table 5. Example of command structure with POI color information.

Command structure	
trigger:	'OK CAR'
task:	'GO'
location:	'THERE'
object:	' '
POI_x:	5.4030
POI_y:	-8.8726
POI_color:	'yellow'

E. Summary

This chapter focused on answering the *Research Question 1* of this dissertation. First, a method for POI computation when the target POI is out of the sensors range was developed. This method was based on the intersections of pointing rays over the time of

the pointing gesture process. In this way, these intersection points were used to calculate the coordinates of the target POI.

A total of 383 simulations were performed: 166 for non-cluttered conditions (i.e., pointing to the first building in the field of view without any building in front), and 217 for cluttered conditions (i.e. pointing to a building behind the first building in the field of view).

The simulation results showed that the distance error of this method with respect to a ground of truth was $2.67 \pm 2.45\text{m}$ for non-cluttered scenarios and $4.10 \pm 2.90\text{m}$ for cluttered scenarios. In addition, this method was more accurate than the traditional POI computation based on pointing rays, improving the total accuracy by 36.25%.

13 valid data collection experiments were performed by the author of this research: 7 of them for non-cluttered environments, and 6 for cluttered environments

The limited number of real world experiments showed that the proposed method was 16.25% more accurate than the traditional POI computation based on pointing rays. However, due to current instrumentation limitations in the data collection experiments, this accuracy differs from the one resulting from the simulation results. Moreover, real world experiments may be subject to more sources of noise and inaccuracies. For example, small differences in pointing angles may result in large lateral errors when the distance to the target POI increases. In Chapter I it was assumed that the car would be equipped with the needed instrumentation elements and HUD technology to provide the proper visual feedback to the user. This would mitigate the pointing issues and assist the

user during the pointing process. Moreover, in Chapters VI and VII a dialog feedback method will be presented to potentially solve these POI ambiguous results.

The second part of this chapter presented a POI computation method for target POIs that are in the range of the sensors of the car. The simulation results showed that this method was able to identify the target POI correctly, with a distance error with respect to the ground of truth of $0.95 \pm 0.09\text{m}$ for non-cluttered conditions and $1.24 \pm 0.28\text{m}$ for cluttered ones. The presented simulation was based in the information provided by a vision sensor with depth filter as described in Chapter IV. However, in more complete scenarios, this information could be complemented by using the point cloud formed by other vehicle sensors, such as LIDAR or radar.

The system developed in this dissertation is based on voice and pointing gestures. Once the POI challenge has been solved, next chapter analyzes the role of voice commands in the command structure creation and in the dialog feedback system to the user for solving potential command conflicts and keep the user informed about them.

VI. CHAPTER SIX: THE ROLE OF VOICE TO CREATE THE COMMAND SEMANTICS AND DIALOG FEEDBACK TO THE USER

A. **Introduction**

The other component of the command structure described in Chapter IV is the voice part of the command. This part has the objective of complementing the POI computation in such a way that, depending on the command semantics, the vehicle can perform the corresponding tasks.

Chapter III described how some research related with the control of the steering, acceleration and braking of an autonomous car by means of different interaction techniques such as eye tracking [92] or brain control interfaces (BCI) [93] was performed. These interaction modes were identified as exhausting by the researchers. In addition, no real control to the user is given: these approaches do not allow the user to make spontaneous decisions over the autonomous route, more than steering, accelerating or braking. Moreover, these commands did not analyzed the role of their semantics in the driving environment and the feasibility of the command in terms of traffic rules compliance or safety is not analyzed. The development of the system of this dissertation requires to identify a target POI in the vehicle's surroundings (analyzed in Chapter V) and process the semantics of the control command in order to update the corresponding autonomous path and evaluate its feasibility in terms of safety and traffic rules. This makes the development of this system a more complex problem than the research referenced previously.

In the field of mobile robots guidance and robots teleoperation, research has shown how to calculate target POI using pointing gestures and voice commands, so the robot can move towards it [95]-[97]. However, in these studies, the human is usually facing the mobile robot and the testing scenario is well-known and structured. In the case of the research in this dissertation, the human operator is inside the vehicle-robot, adding complexity to the problem.

Dialog systems based on natural language have been used in the automotive field to communicate with drivers in a natural way [83]-[86]. However, this research is mainly focused on secondary functions of the vehicle. In the field of robotics, dialog feedback is also used, for example, to teach the robot new tasks [101], or to give directions to a robot [141] in structured scenarios. The presented research involves a more complex scenario, where a dialog system must provide means to disambiguate potential POI or commands conflicts, before a command can be executed.

This chapter elaborates on the role of voice interaction as a means to provide semantics to the command as well as establish a dialog feedback system to solve potential conflicting situations. In particular, it answers the **Research Question 2: *How can voice and pointing gesture commands be combined to form a semantically complete command to the autonomous vehicle and give the proper feedback for conflicting situations such as POI identification ambiguity or not feasible user command?***

The first part of this chapter presents a proof of concept of a system that combines voice and pointing gesture commands.

Afterwards, this system is further developed in the simulation framework described in Chapter IV. This includes a dialog feedback system to disambiguate potential conflicting situations. As it will be described in Chapter VII, an adaptation of this feedback system also improves the accuracy of the POI computation presented in Chapter IV.

B. Command structure implementation

Based on the command structure defined in Figure 5 of Chapter IV, this part of the chapter implements it. First, a proof of concept of the system, depicted in Figure 32, is implemented. Second, this system is developed using the simulation framework described in Chapter IV.

1. Proof of concept

The objective of this proof of concept is to showcase how a user can command an autonomous car by means of a command structure based voice and pointing gesture commands. However, to be able to implement it, several building blocks of the system architecture described in Chapter IV were implemented. In particular, those red-dotted in Figure 32.

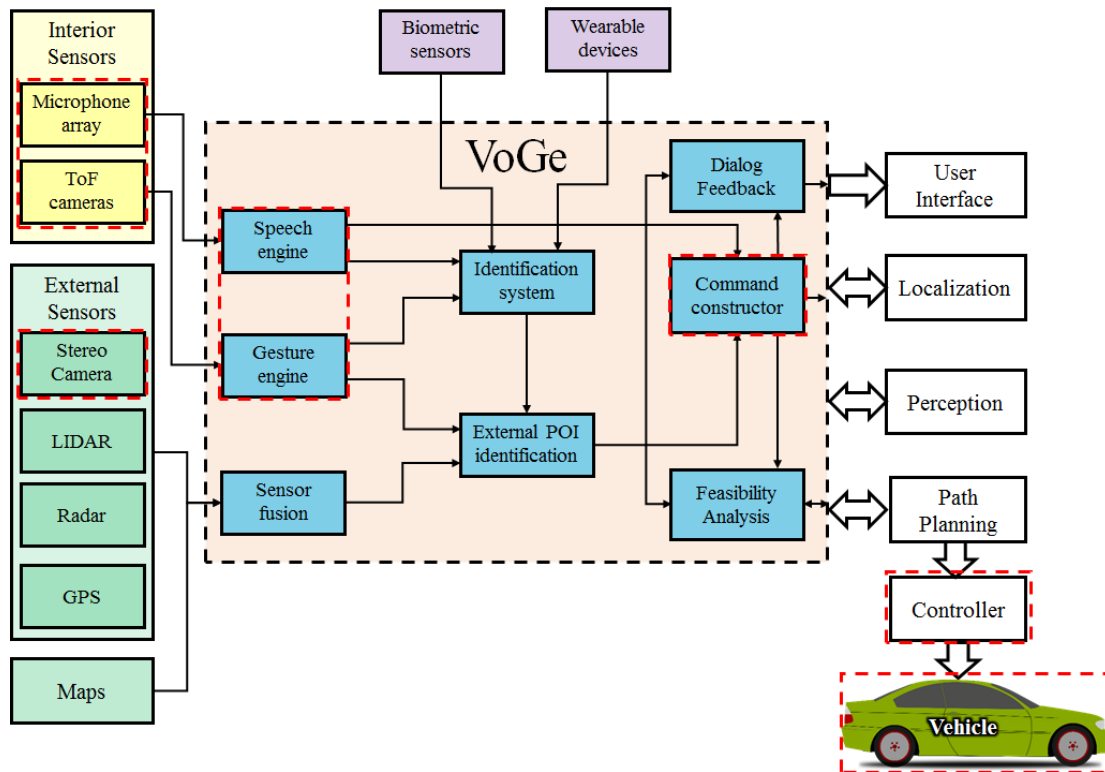


Figure 32. Implemented modules in the proof of concept.

In this case, the prototype was set up in small scale, using a driving simulator laboratory and a robot that acted as the autonomous car (Figure 33).

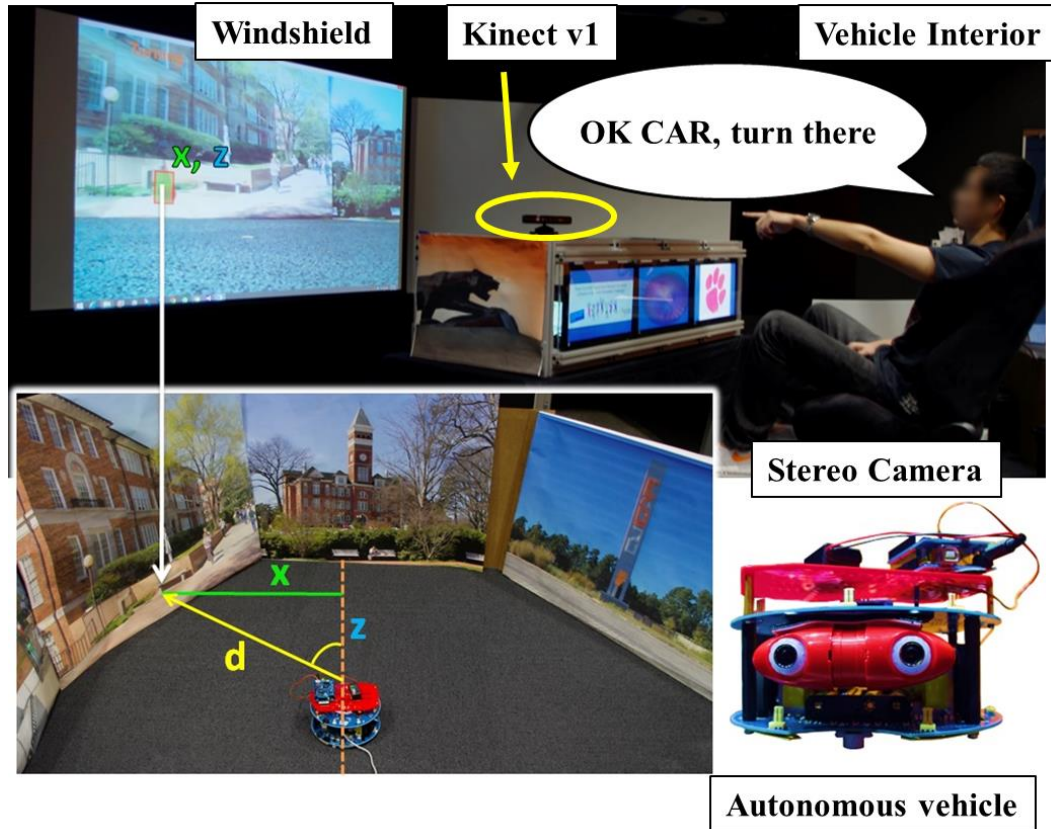


Figure 33. Proof of concept set up.

The main parts of the prototype are described in the following subsections.

a. Vehicle Interior mock-up

The main objective of the vehicle interior mock-up is to simulate how a user would interact with the voice and pointing gestures system inside a car without any control elements such as steering wheel and pedals.

The vehicle interior mock-up counts with a dashboard concept frame without steering wheel, designed and constructed as a hollow box that can hold displays and other

HMI (Human Machine Interaction) devices. This platform provides the flexibility needed to implement and test new HMI concepts in the future. For this prototype, three displays were set up.

A moving seats platform in combination with the dashboard provides an open layout to further test the interaction flexibility of the voice and pointing gesture commands in future prototypes.

The multimodal interaction interior sensors are formed by a Microsoft Kinect Sensor. This, along with the Microsoft Kinect SDK [126] provides the capability of prototyping multimodal interactions based on voice and gestures.

The target of these pointing gestures is a projector display that acts as the vehicle's windshield. The robot streams video to the display, so, as it will be explained, the user points to it in order to identify the target POI.

Finally, the 'vehicle intelligence processor' is formed by a computer which receives the streaming images from the robot, process the multimodal voice and gesture commands, and transmits them to the robot. The streaming images are captured using EmguCV [142]: a C# wrapper of the computer vision library OpenCV [143].

As it was already mentioned, the robot acts as the 'autonomous vehicle mockup', which is explained in the next subsection.

b. Autonomous Car Mock-up

A robot based on the Arduino platform [144] was used as the autonomous car mock-up. It was selected because every element in the platform was freely available,

open-source and it facilitated the prototyping of the system. This robot counts with a low-cost stereo camera (Minoru 3D [145]) which acts as the autonomous car external sensor. The images captured by this camera are streamed to the vehicle processor (a desktop computer in this case). In this way, when the user points to a point of that image, a POI is calculated and the corresponding command is transmitted via a wireless connection to the robot. Due to latency issues, the images from the stereo camera are transmitted via USB instead of using a wireless link. However, in a real car implementation, this transmission would be done via the buses of the vehicle. Thus, this is not considered a major issue for the prototype implementation, which is explained in the next subsection. Moreover, the user of the depth information provided by the vision sensor was studied in Chapter V and used in our simulation framework.

c. Voice and pointing gesture system prototype implementation

As it was explained, a full implementation of the proposed system requires a set of inputs, software implementation and interaction with other systems/controllers of the car. For this stage of the research, we showcased the red-dotted blocks of Figure 32.

A wireless link between the ‘vehicle processor’ (ECU) and the robot (autonomous car) is created in order to exchange data between them. In a real world scenario, this link would be created through the buses of the car. The robot is continuously sending to the processor images of the environment captured by its stereo camera. Due to the high latency experienced, this link was finally done through a USB connection. These images are displayed in the projector, which simulated the windshield of a real car. In addition,

acting as the sensor fusion block of Figure 32, a layer of stereo image is processed using a block matching algorithm [146]. In this way, when a user is pointing to a pixel in the projected image (the simulated car's windshield), the information of that point (pixel position) will also include the distance of the robot to the pointed object (depth Z of the pixel). Using this POI position information, we can later calculate a triangle which will determine the heading direction of the robot as well as its distance to the POI. This is the prototype version of the external POI identification module. A proper version of this module was described in Chapter V related to ***Research Question 1***.

The stereo vision algorithm was implemented using EmguCV [142]. However, at the time of the proof of concept implementation, the disparity maps provided by the stereo vision algorithm had considerable noise. This prevented to use them in the proof of concept. However, as we have just mentioned, this is covered in Chapter V.

In order to capture the voice and pointing gestures of the user, the speech and gesture engine modules were developed using Microsoft Kinect SDK [126]. A grammar was created in order to correlate the voice signals with the final command structure depicted in Figure 5. The system is constantly monitoring audio signals in order to detect the trigger command «OK Car», which is also defined in the grammar. When this command is identified, the rest of the command structure is analyzed and the gesture engine detects the human body joints in order to form the pointing vector. This information is used by the external POI identification module to calculate the target location the user is pointing to.

Finally, the command constructor module creates the command that will be transmitted to the robot. Based on triangulation calculations direction, angle and distance to follow are transmitted to the robot. The action to perform is also transmitted. These are some command use cases for this proof of concept:

- «*OK car + stop + pointing gesture + there*»
- «*OK car + take + pointing gesture + that exit*»
- «*OK car + turn + pointing gesture + there*»
- «*OK car + go + pointing gesture + there*»

This proof of concept, which overall sequence diagram is shown in Figure 34, shows how a command structure based on voice and pointing gestures can be implemented in order to communicate on-route spontaneous decisions to an autonomous car. Thus, being an important outcome to answer ***Research Question 2***.

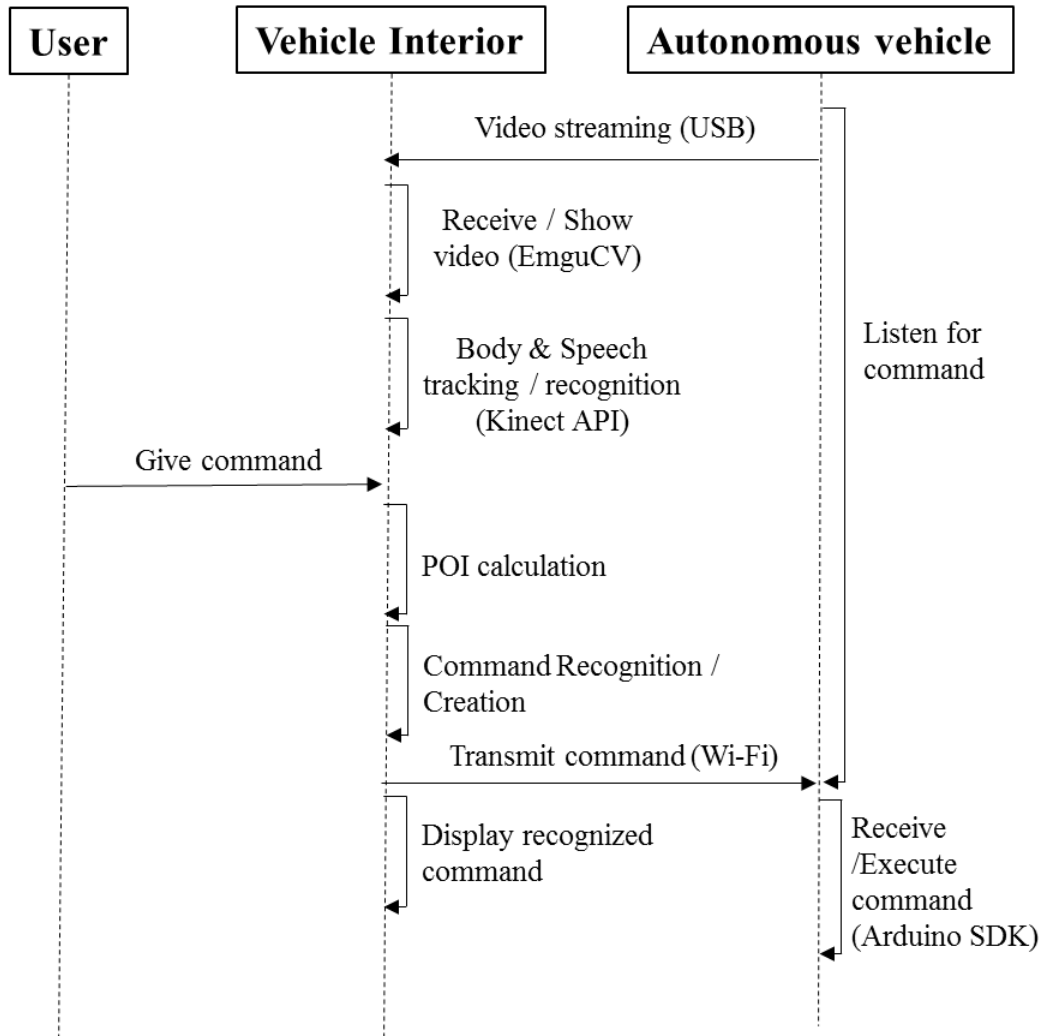


Figure 34. Proof of concept sequence diagram.

2. Voice Commands in simulation framework

The voice and pointing gesture command structure to ‘command’ an autonomous car presented in this research was showcased in the proof of concept described in the previous subsection. However, this did not include any path planning and path following. Thus, the focus of this section of the dissertation is to integrate the voice and pointing

gesture information in the same command structure, so that the vehicle can perform the corresponding command by updating its pre-defined autonomous path.

a. Speech recognition engine design

The *speech recognition engine*, developed using the library *System.Speech* [121] makes use of custom built grammars that were created for this dissertation. In this way, when a user performs a voice command, this is compared against the corresponding vocabulary, depending on, as we will describe in Chapter VII, what is the state of the system. In particular, the following grammars were developed using the *W3C Recommendation “Speech Recognition Grammar Specification”* [122].

1) User command grammar

This grammar (Figure 35-Figure 39) is used for recognizing the voice part of the command structure described in Figure 5. In this way, its semantic output combined with the target POI information results in the final user command structure. Although in a final implementation the voice and pointing gesture parts of the command can happen in parallel (similar to the work presented in [147], [148]), this dissertation presents them in sequence for simplification purposes. For example, the voice command: “*Ok car, could you please go there*”, will recognize “*OK car*” as the trigger, “*go*” as the task and “*there*” as the *location*. In this case, “*could you please*”, is the GARBAGE part of the command, i.e. any spoken token that should be ignored by the speech recognition engine.

In addition, the object part is optional. In this way, natural voice commands can be performed and recognized. An example of this process is depicted in Figure 40.

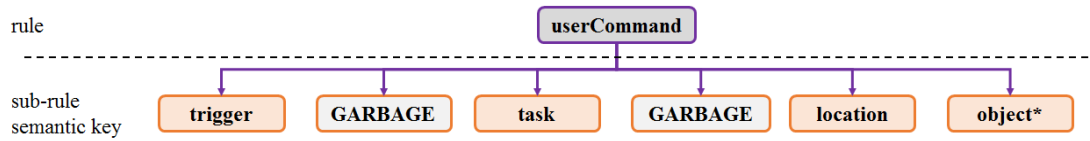


Figure 35. User command grammar overview.

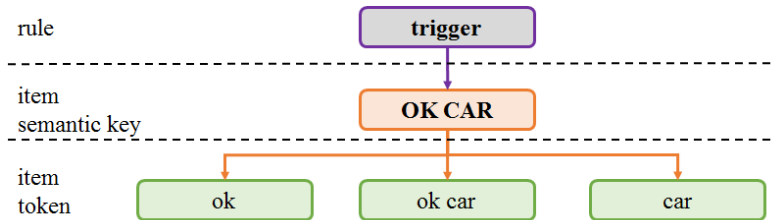


Figure 36. Trigger part of user command grammar.

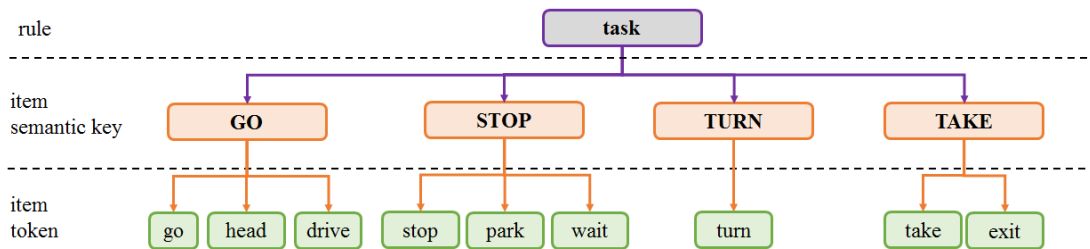


Figure 37. Task part of user command grammar.

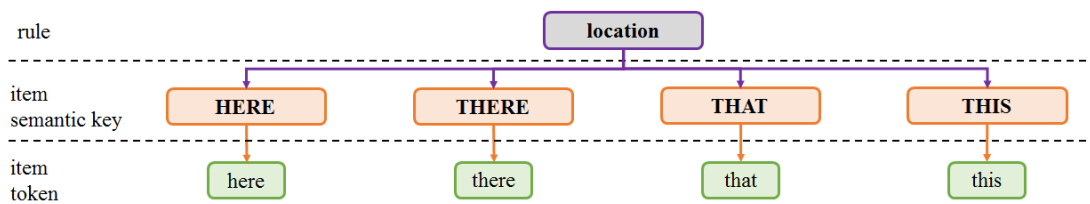


Figure 38. Location part of user command grammar.

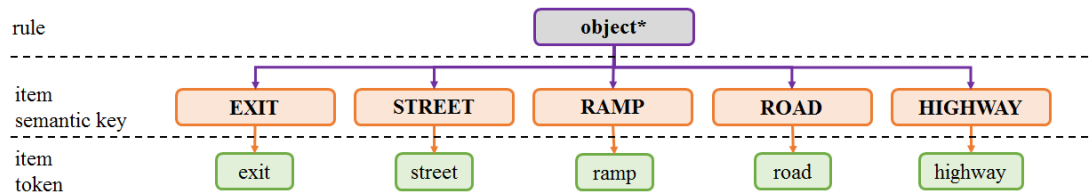


Figure 39. Object part of user command grammar.

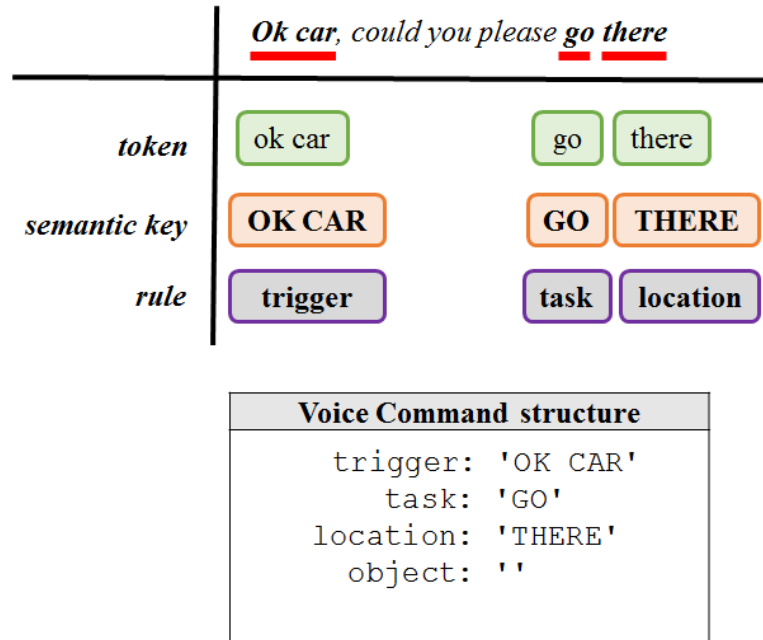


Figure 40. Speech recognition process example.

2) Colors grammar

This grammar (Figure 41) is used as part of the dialog feedback system module.

In case of target POI ambiguity, via the dialog feedback system the user will be able to specify the target POI he/ she is referring to.

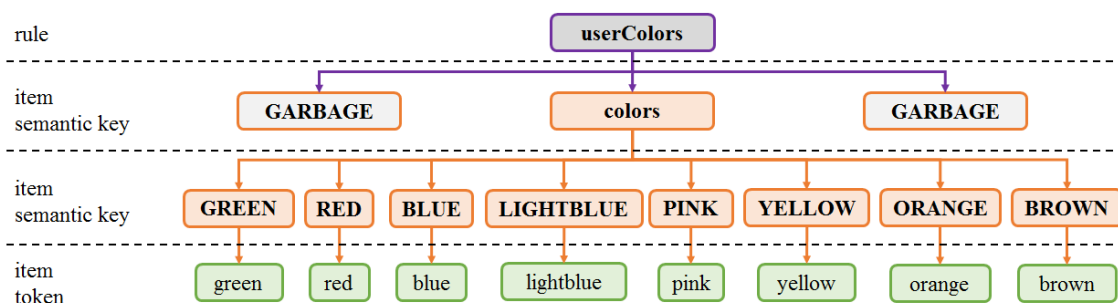


Figure 41. Colors grammar.

3) Confirmation/ rejection grammar

This grammar (Figure 42 and Figure 43) is used to either confirm or reject the command performed by a user. As it will be described in Chapter VII, if the user rejects the command, he/ she can do it by directly rejecting it, reject it by color (the POI computation module failed to detect the correct target POI), by task (the voice recognition module misunderstood the task semantics), or by color and task (POI computation and voice recognition failed). In these last cases, the user would be able to update the color and / or task of the corresponding command, correcting it. It is important to note that the task and color parts of the rejection part of this grammar (Figure 43) are optional and have the same structure of the grammar parts of Figure 37 (for task) and Figure 41 (for color).

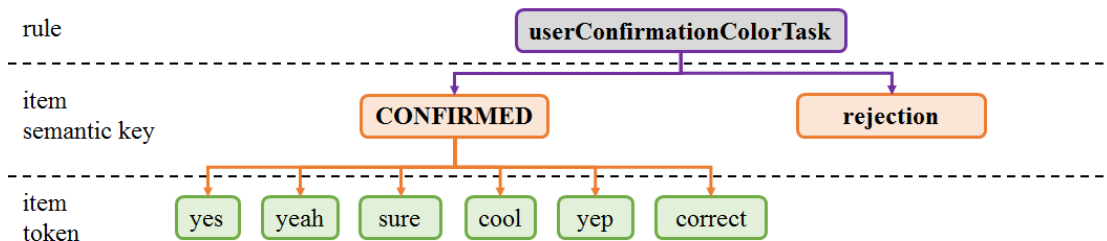


Figure 42. User confirmation/rejection grammar overview.

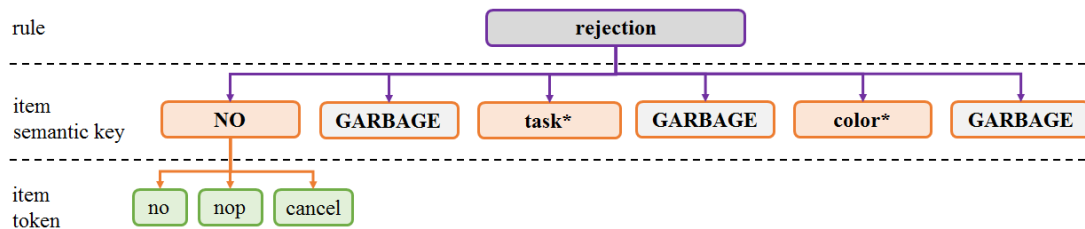


Figure 43. Rejection part of the confirmation/ rejection grammar.

b. Command constructor module

In Chapter V we described how the POI computation module calculated the coordinates of the target POI. In this chapter, the voice recognition module has been described.

Once the target POI coordinates are calculated, the system combines them with the information extracted from the user's voice command part. Table 6 shows an example of the structure created from the command «OK car, go there», where POI_x and POI_y are the real world coordinates of the pointed POI with respect to our 3D simulation environment coordinate system and POI_color is de detected POI characteristic of the target POI.

Table 6. Example of command structure.

Command structure
trigger: 'OK CAR'
task: 'GO'
location: 'THERE'
object: ''
POI_x: 5.4030
POI_y: -8.8726
POI_color: 'yellow'

As it was already mentioned during this dissertation, some challenging situations may happen when the user performs a command. Such is the case of voice command misunderstanding, POI computation ambiguity or command not feasible in terms of safety and traffic rules compliance. For this reason, a voice based dialog feedback system needs to be developed. In this way, by means of a user-car dialog, the challenging situation can be solved. The next subsection explores this dialog feedback module.

C. Dialog feedback system

The second part of *Research Question 2* consists on the implementation of a dialog feedback system to help to solve conflicting situations such as POI ambiguous computation or not feasible command in terms of safety and traffic rules compliance. As it will be explained in Chapter VII, the dialog feedback system to the user is based on the different states outputs of the level structured FSM that constitutes the whole system presented in this dissertation. In this way, depending on the system state, different feedback messages and voice recognition grammars (such as the ones described in this chapter) can be enabled and disabled. Figure 44 shows the basic process of the feedback creation. Depending on the system states or on other system events, a message is formed and transmitted to the systems voice synthesizer so that the voice feedback can be generated.

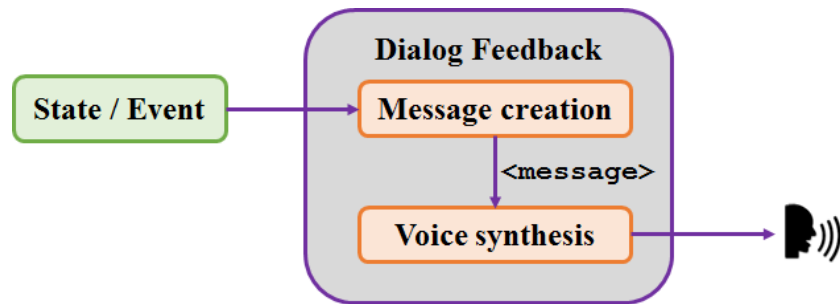


Figure 44. Dialog feedback creation

This dialog feedback module, combined with the speech recognition module, help the car (system) and the user to collaborate to solve potential conflicting situations.

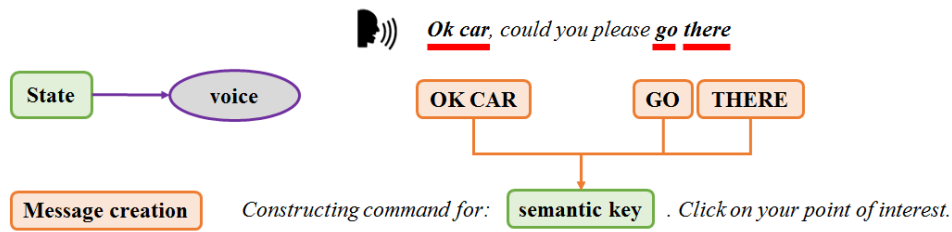
In this dissertation the following dialog feedback use cases are explored:

1. Voice command not recognized

When a user performs a voice command and the speech recognition engine cannot recognize it with enough confidence level as part of the enabled speech recognition grammars, the car asks the user to repeat it by giving the following voice feedback: I did not understand it. Could you repeat it? In this way the user could repeat again the intended voice command and is aware of the system state of command misunderstanding.

2. Voice command recognized

When the voice command performed by the user is recognized with enough confidence level, the car informs of this situation to the user, so that he/ she can perform the pointing gesture. Figure 45 shows how the feedback message is formed based on the recognized command semantics. In addition, as it will be described in Chapter VII, this feedback is triggered when the system is in the 2nd level state *voice* and no command has been recognized yet. It is important to note, as it was stated in Chapter IV, that the voice and pointing gesture part of the command can happen in parallel. In this case, this message would be triggered if no pointing command is detected during the command recognition part. In this dissertation, for demonstration purposes, this process is structured as a sequence; thus, this feedback is triggered every time a voice command is recognized.



Constructing command for: OK CAR GO THERE . Click on your point of interest.

Figure 45. Command recognized feedback.

3. POI disambiguation

It is possible that the POI computation module detects more than one potential target POI. In this case, the system will ask the user what would be the correct POI, taking into account a POI property. In this case, color was used as the main POI property. However, other could be added (in addition to add them to the corresponding voice recognition grammar). For example, as it is depicted in Figure 46, the car may ask to the user Are you referring to the **blue** or **red** building? because it detected two potential target POIs. In this way, the user can tell (using voice) the correct building color and the system will search it in its potential target POI list and set the correct target POI coordinates. The color voice command selected by the user is processed in the speech recognition engine by checking the Colors Grammar described previously in this chapter.

As it will be described in Chapter VII this feedback is triggered when the system is the 3rd level state *waiting_poi_color* (in general, for each POI property to be confirmed there should be a state). If the user and the vehicle dialog reach to a valid POI solution,

the system will transition directly to the 4th level state *command_confirmed*. If the user specifies a color that was not in the potential POI list, the system will notify to the user with the following message: Sorry, the color was not in the list. Waiting for new command. In that moment, the system will wait for a new command while the vehicle is following the original path.

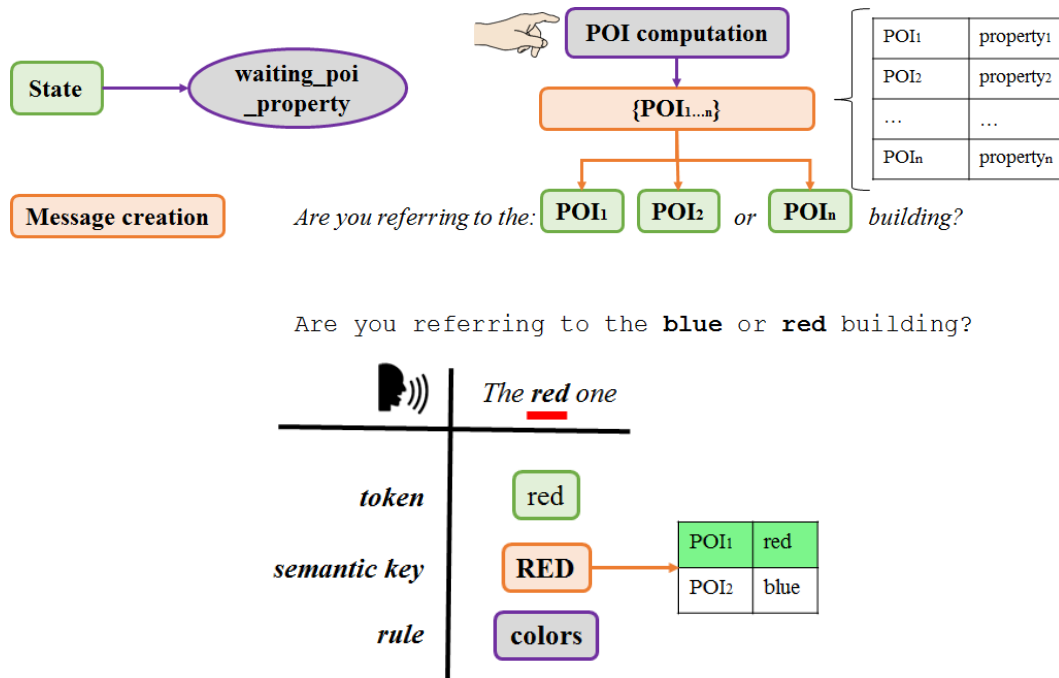


Figure 46. POI disambiguation dialog.

A simplified case for this feedback is the one related with *Research Question 1*. In order to solve POI computation inaccuracies presented in Chapter V, the system only prompts a message informing the POI that has been detected, being its properties color or name, depending if the feedback is provided for simulation or real world experiments.

That is, in these particular cases the system does not provide voice feedback, because the scope was to determine if an additional confirmation step was able to solve the POI inaccuracies. This will be further described in Chapter VII.

4. Command confirmation request

In the case the system detects a single potential target POI, it will ask the user for confirmation. In this way, the user is aware of the command the vehicle intends to perform and can confirm, reject or correct it. Figure 47 shows an example of this. When the system creates a temporary command, it asks the user for confirmation: Did you mean **GO** to the **yellow** building? As it will be described in Chapter VII this message is created when the system is in the 4th level state *waiting_confirmed*.

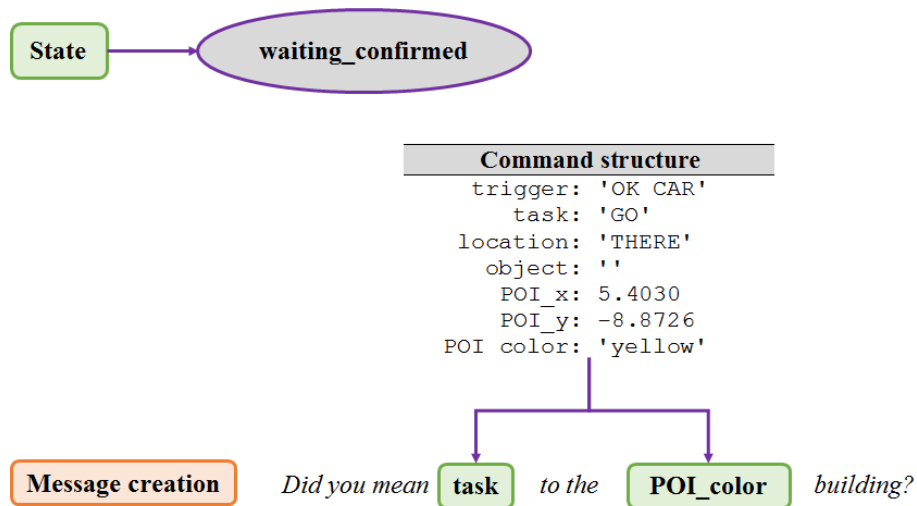


Figure 47. Command confirmation feedback.

5. Command confirmation / rejection / correction by the user

Once the system asked for command confirmation, the user can either confirm, reject or correct (color, task or both) the command. If the user corrects the command (for example, No, the **blue** one) the system will ask one last time for confirmation (Did you mean **GO** to the **blue** building?). If the user rejects the command this second time, the car will start to listen for new commands again while following the pre-defined autonomous path. The reason for this limited number of confirmation requests is to avoid infinite dialog feedback events. Figure 48 shows some examples of this case. Chapter VII will describe the command update process depending on the user feedback response. In addition, this message is created when the system is in the 4th level state *waiting_confirmed*.

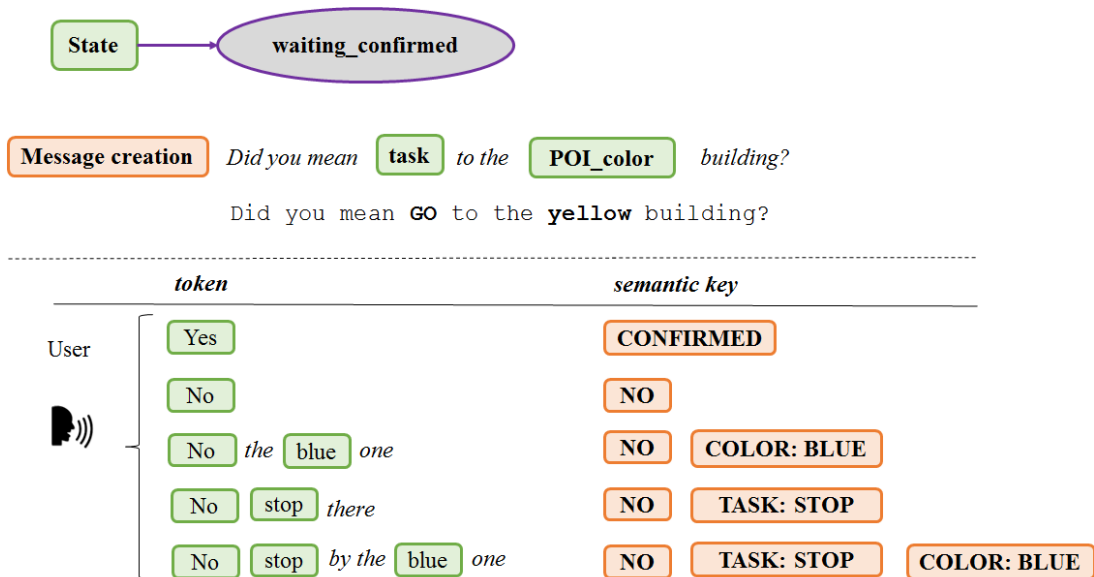


Figure 48. Command confirmation, rejection, and correction dialog.

If at the moment of the command correction process the user wants to update the detected POI color and this is not in the field of view of the vehicle, the system will generate the corresponding feedback and will wait for a new command: I cannot find the **<updated_color_name>** building. Waiting for new command. If the color is in the vehicle's field of view, color filtering functions available in the computer vision toolbox used in this dissertation [123] are applied, so that the correct building can be identified and its coordinates calculated (based on the centroid of the filtered building area).

If the user confirms the command, the following feedback will be generated: Command confirmed. Executing. Waiting for new command. This feedback is triggered when the system is in the 4th level state **command_confirmed**.

On the other hand, if the command is rejected, the system will generate the following feedback message: Command rejected. Waiting for new command, and will reset its states to listen to new commands, as it will be explained in Chapter VII.

6. Feasibility analysis feedback

As it will be described in Chapter VII, the feasibility of the command it is analyzed in terms of traffic rules compliance (i.e. we assume that if the command does not comply with traffic rules, it is not safe). In this way, if the result is negative, the system will give the proper feedback to the user. Figure 49 shows this process. As it is

shown, the message is formed taking into account the task part of the command and the conflicting traffic situation resulting from the feasibility analysis module.

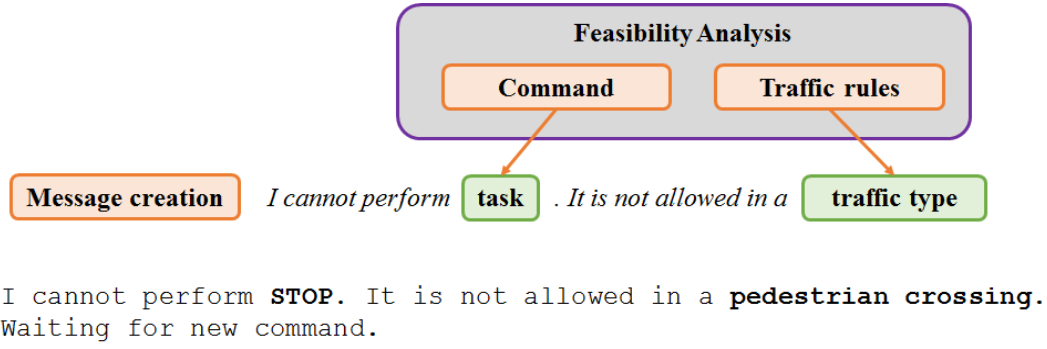


Figure 49. Feasibility analysis feedback.

D. Additional system modules

To complete the system simulation framework described in Chapter IV, the following additional modules were implemented. This subsection shows as well how the pre-defined autonomous path can be modified depending on the semantics of the user command.

1. Path following module

In order to make the vehicle follow the defined path, a **PID** controller was implemented following the method described in [149]. Although other path following approaches are possible [150], [151], the one used in this research was chosen due to its simplicity.

Considering a vehicle defined by its bicycle model and a matrix of path points $P \in \mathbb{R}^{n \times 2}$; the scheme of Figure 50 shows the relations between the vehicle position and the segments of the track to be followed.

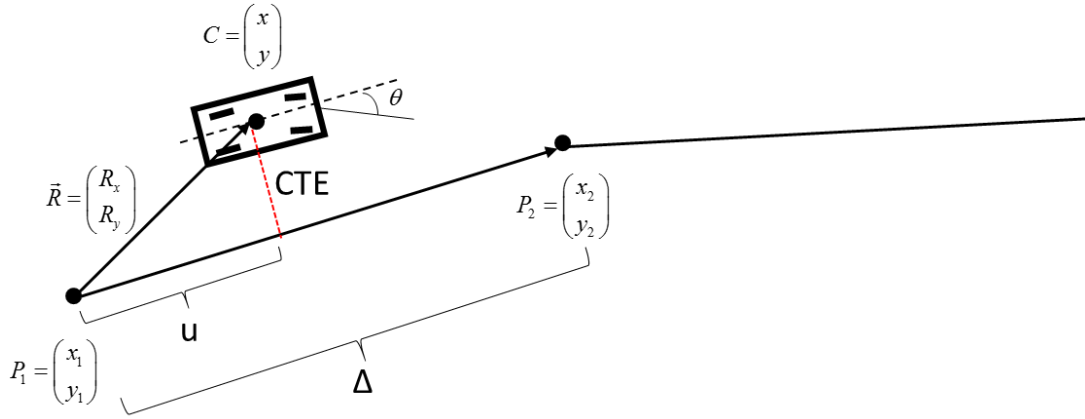


Figure 50. Path following overview.

The Cross Track Error (CTE) (i.e. lateral error) can be defined as:

$$CTE = \frac{R_y \cdot \Delta x - R_x \cdot \Delta y}{\sqrt{\Delta x \cdot \Delta x + \Delta y \cdot \Delta y}} \quad (11)$$

Where,

$$\begin{aligned} R_x &= x - x_1 \\ R_y &= y - y_1 \\ \Delta x &= x_2 - x_1 \\ \Delta y &= y_2 - y_1 \end{aligned} \quad (12)$$

The steering angle θ needed to minimize the CTE is given by:

$$\theta = -K_p.CTE - K_D.\frac{d}{dt}CTE - K_I.\sum CTE$$

$K_p \rightarrow$ Proportional gain
 $K_D \rightarrow$ Differential gain
 $K_I \rightarrow$ Integral gain

(13)

For this particular case, using the heuristic method, $K_p=2.5$, $K_D = 5.5$ and $K_I = 0$ were considered.

If $u > 1$, the vehicle is switching from a path segment to the following one; thus, changing the set of path points used in the controller. This variable can be defined as,

$$u = \frac{R_x.\Delta x + R_y.\Delta y}{\Delta x.\Delta x + \Delta y.\Delta y}$$
(14)

In addition, for this particular case, a constant speed of 3 m/s was considered.

As it was described in Chapter IV the execution of this controller involves a synchronization between processing and 3D simulation environments. While the 3D side provides the vehicle position and orientation at each simulation time, the processing side sets the calculated steering angle θ and velocity (a constant value of 3 m/s in this case), so the vehicle in the 3D environment side can **move**. This vehicle follows a simplified version of the Ackermann car model [118] shown in Figure 51.

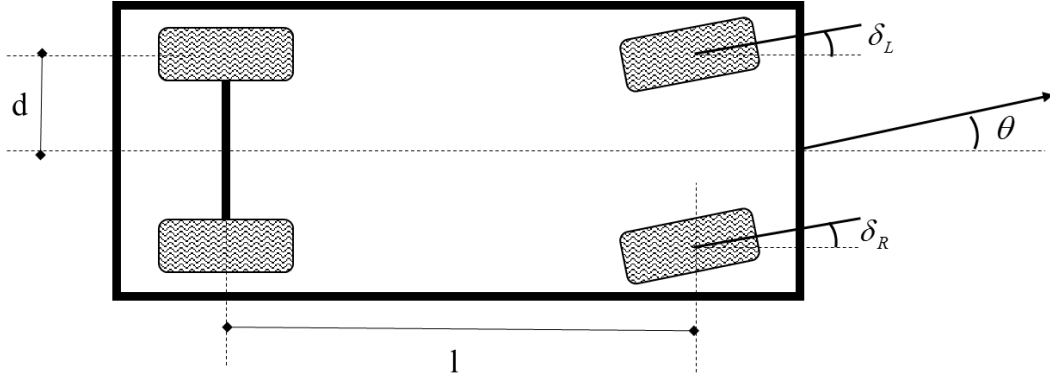


Figure 51. Simplified Ackermann model.

Where,

$$d = 0.755\,m$$

$$l = 2.5772\,m$$

$$\theta \in \left[-\frac{\pi}{4}, \frac{\pi}{4} \right] \text{ rad}$$

$$\delta_L = \tan^{-1} \left(\frac{l}{-d + \left(\frac{l}{\tan(\theta)} \right)} \right) \text{ rad} \quad (15)$$

$$\delta_R = \tan^{-1} \left(\frac{l}{d + \left(\frac{l}{\tan(\theta)} \right)} \right) \text{ rad}$$

Using this approach, the autonomous vehicle can follow a defined path, which is a needed step to be able to simulate the system presented in this dissertation.

2. Path update module

Once the user confirms the command, a new path is calculated based on the target POI coordinates and the task indicated in the command.

In this way, given the original path $P_{original} = \{C_{start}, ..., C_{end}\}$, where C_{start} and C_{end} are its *start* and *end* coordinates, two options are possible. When a user performs a GO THERE or STOP THERE command, a new path is calculated by the path planning module of the 3D environment side in such a way that $P_{new} = \{C_{car}, ..., C_{POI}\}$, where C_{car} is the current car position and C_{POI} are the coordinates of the target POI. Figure 52a shows an example of this case. The vehicle starts to follow its original path (red line) in the ‘Start’ position. While the red diamond corresponds to a calculated POI of a command that was rejected, if the user performs and confirms another command, a new path (blue line) is created and followed (green dotted line) until the vehicle reaches the new POI (end of path).

In the case a user gives a TAKE THAT EXIT or TURN THERE command, the new calculated path has the form $P_{new} = \{C_{car}, ..., C_{POI}, ..., C_{end}\}$. In this way, the car navigates towards the target POI coordinate C_{POI} and continues its path to the original path end coordinate C_{end} . Figure 52b shows an example of this case. For this particular scenario, the calculated POI (black diamond) is the landmark where the vehicle has to turn, continuing afterwards to the original final destination.

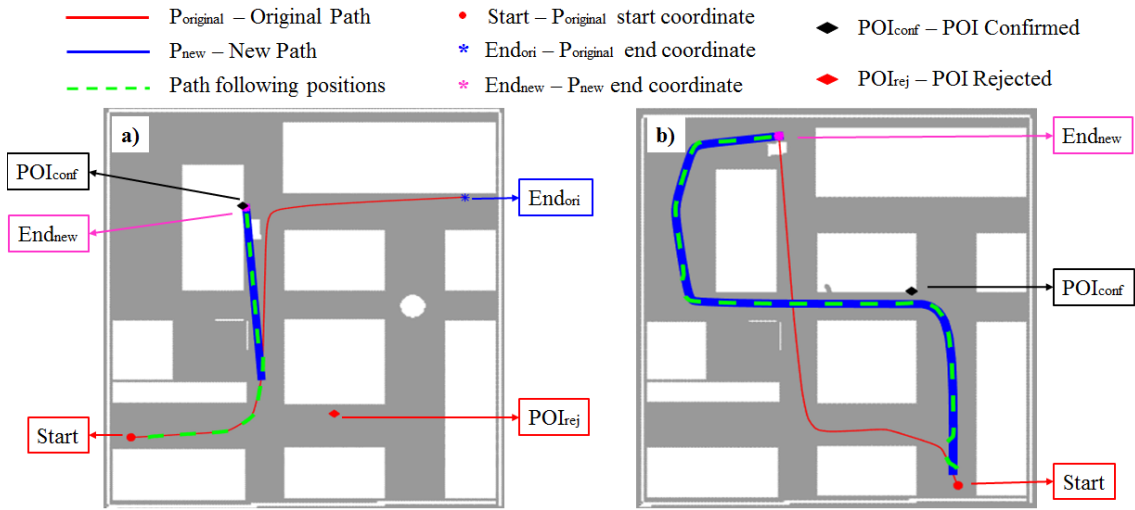


Figure 52. Examples of path updates: a) 'Go there' command; b) 'Turn there' command

The basic scheme of the new path landmarks formation depending on the command semantics is depicted in Figure 53.

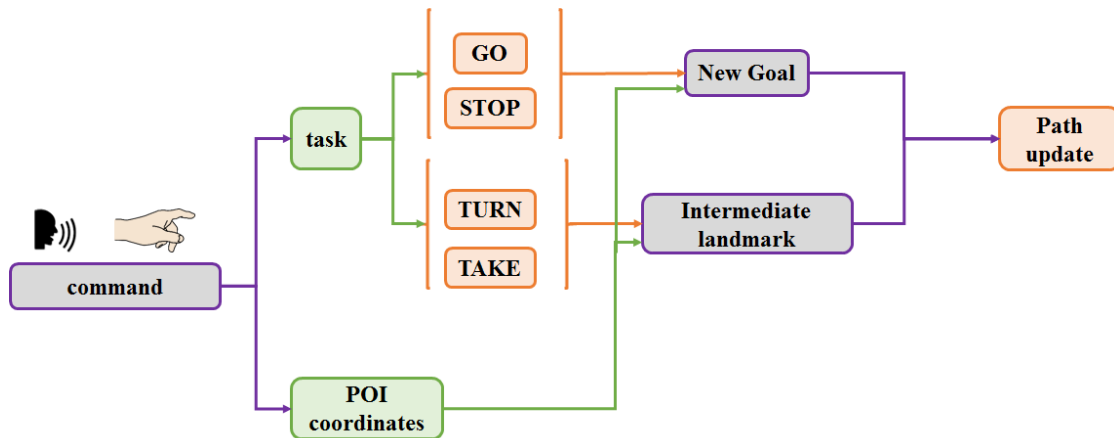


Figure 53. New path landmarks creation depending on command semantics.

Finally, the result the path planning module is a matrix of path points $X \in \mathbb{R}^{nx2}$.

These path points can be converted into a ‘drivable’ **smooth path**, resulting in a

smoothed path points matrix $Y \in \mathbb{R}^{nx2}$ such that [149],

$$y_i = y_i + \alpha.(x_i - y_i) + \beta.(y_{i-1} + y_{i+1} + 2y_i)$$

where,

data coefficient: $0 \leq \alpha \leq 1$

smooth coefficient: $0 \leq \beta \leq 1$

(16)

E. Results

The simulation framework described in Chapter IV was executed in order to analyze the performance of the speech recognition engine and dialog module of this chapter.

A total of 84 experiments combining all the voice recognition grammars and dialog feedback events explained in this chapter were performed.

The speech recognition engine reached an overall combined accuracy rate of 79.5%. A major factor (88.5%) that influenced the potential inaccuracies in the speech recognition process was the inclusion of the GARBAGE items (words) of the grammars described in this chapter. The objective of this GARBAGE is to make the speech process more natural to the user. For example, a user may find more natural say “*No, I meant the red building*” than “*No, red*”. The speech recognition engine, then, filters the GARBAGE and only recognizes the important parts of the sentence (“*No*”, and “*red*” in this case). However, it was found that this affected the accuracy of the recognition process, as the GARBAGE parts may be confused by actual words that the speech

recognition system has to recognize. Other sources of voice recognition inaccuracies are related with words that may have similar pronunciations (for example, *green* and *grey*). Thus, as it will be mentioned in Chapter VIII other speech recognition engines may be used in future implementations.

In Chapter IV the system command structure was described (Figure 5). The creation of this command structure by the speech recognition engine and the gesture recognition engine lead to the successful creation of the corresponding command. If the command was confirmed by the user, the new path was always successfully calculated and followed, as depicted in Figure 52.

Experiments were also performed to test the POI disambiguation and Command confirmation / rejection / correction dialog events, and how the system recognized and updated the corresponding command structure.

In the case of POI disambiguation dialog, presented in Figure 46, when the system detects more than one potential POI, the system was able to find the correct POI in its potential POI list every time the semantics of the user reply to the system feedback were recognized (80%), and calculate and follow the new created path.

In the case of command confirmation / rejection / correction dialog, the semantics recognition corrected colors lead to a correct new POI calculation 73% of the times. The remaining times, the functions of the computer vision toolbox used in this research [123] had issues to filter the correct building color. This could be solved by adding more color RGB codes to the corresponding functions and is not considered a problem in this dissertation, as its objective is not to provide a color filtering system, just make use of it.

However, adding more colors, means add them to the colors grammar of Figure 41, adding more complexity to the recognition process. By using annotated maps, as the one described in Chapter V, with color as a map element property, the use of computer vision to disambiguate the potential POI can be avoided. This will be explained in Chapter VII. If the user corrected the task part of the command, this was properly updated all the times the semantics of the task were recognized.

Finally, all the command structures created by the combination of voice and gestures, and the dialog feedback messages were successfully created for all the simulations. Moreover, the dialog feedback module allowed the user to be aware every time of the system status.

F. **Summary**

This chapter focused on answering the *Research Question 2* of this dissertation. First, a proof of concept was developed using a set up based on a driving simulator, an Arduino robot platform [144], a low cost stereo camera [145], and a Microsoft Kinect sensor [126] (Figure 33). This proof of concept showcased how a user could command an autonomous car by means of a command structure based on voice and pointing gestures.

The second part of this chapter focused on the development of the voice recognition engine and the associated speech recognition grammars. In this way, the semantics of the user voice commands were combined with the target POI coordinates computed in *Research Question 1* to create the system's command structure. This command allowed to update the autonomous path of the car, depending on the semantics

of the task that the user intended the car to perform. Some final results of this process were depicted in Figure 52.

In addition, other grammars were defined to be recognized when the dialog feedback system module asks the user for command or POI clarification. Of the 84 experiments performed, the voice recognition accuracy was 79.5%, which led to the correct command creation, correction if needed and its corresponding path update.

This demonstrates how a voice based dialog feedback system is needed to solve potential conflicting situations such as negative result of the feasibility analysis module or POI disambiguation, and make the user aware of the system state.

As we already mentioned, the system of this dissertation is designed as a level structured FSM in order to be able to handle any event that leads to its corresponding dialog message to the user. This is analyzed in the next chapter.

VII. CHAPTER SEVEN: LEVEL STRUCTURED STATE MACHINE TO HANDLE CONFLICTING SITUATIONS

A. Introduction

This chapter elaborates on **Research Question 3**: *How can this system be designed so that it can handle conflicting situations such as feasibility of the command or POI ambiguity?*

Chapter VIII described the role of the dialog feedback module to give information to the user about the system state, as well as to solve via a dialog, potential ambiguous and challenging situations, such as command not feasible in terms of traffic rules compliance, ambiguous POI computation, or confirmation / rejection of the user command.

This feedback is the result of the design of the system as a FSM. The proposed system in this dissertation is designed as a level structured state machine, with different state levels. In this way, as it will be described during this chapter, the first level is the system level which controls the behavior of the system as a whole. In addition, other levels control command semantics based on voice, POI calculation and command creation. This structure results in a system that can be considered modular and scalable. In particular, as it will be described, the 2nd and 3rd level could be exchanged by other interaction modes, as long as they keep their basic objective (command semantics representation for Level 2 and POI calculation for Level 3). Finally, a feasibility analysis module controls the feasibility state of the resulting command in terms of traffic rules compliance.

Using the simulation framework described in Chapter IV the functioning and interactions between each state level are demonstrated. In addition, the use of POI disambiguation is showcased for improving the results of *Research Question 1*.

B. States design approach

As it was mentioned, the system developed in this dissertation is based on the combination of different subsystems, which keep their internal states and trigger the state transitions to other subsystems. Designing the system as a level structure state machine allows to modular, scalable and flexible. In particular, as it will be further discussed in Chapter VIII, other interaction modes could be used with this same structure. For example, Level 2 is focused on the command semantics. Although it has been named Voice Recognition States level, it could be seen as a Semantics Recognition States. Thus, as long as the interaction allows to provide command semantics, this level would remain the same and no major changes would need to be applied to them. With regards to Level 3, its objective is to compute the target POI. Although this dissertation focuses on pointing gestures, other modes could be used as long as they allow to calculate the target POI. In addition, color was taken as the POI property for POI disambiguation. However, other properties would be possible. Thus a more general state *waiting_poi_property* instead of just color would be more appropriate. Figure 54 shows the main state levels of the system design as well as the transitions between them.

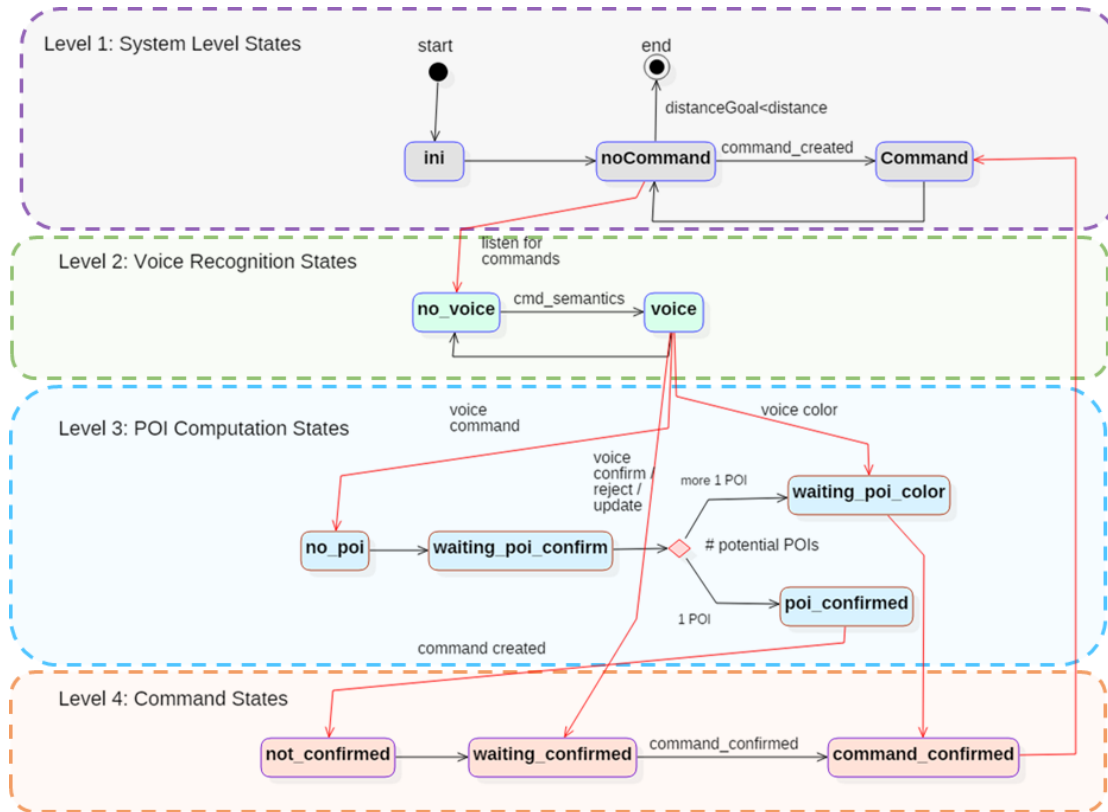


Figure 54. System design as a level structured state machine.

This part of the chapter describes each subsystem state machine and the interaction between them.

1. Level 1: System states

Figure 55 shows the design and main tasks of the system states. This states describe the functioning of the overall system, and it is composed of the following states:

- a) **start** describes the starting of our system, where the synchronization with our simulation framework takes place. Once this synchronization is done, the system transitions to the **ini** state. In addition, all the system states are initialized.

- b) In the **ini** state the system initializes all the vehicle and sensor parameters. In addition, the initial autonomous path calculated by the path planning module is received and smoothed, so it can be drivable. This state also executes the asynchronous speech recognition engine module, which runs in parallel to the rest of the system. Once the initial path is received and smoothed, the system transitions to the **noCommand** state.
- c) The **noCommand** state performs all the main functions of the system. This state triggers the rest of the state machine levels that are described in this chapter. That is, this state can be considered the parent of the rest state machine levels. This state executes the gesture recognition engine, sensor fusion, external POI identification, command constructor, dialog feedback and feasibility analysis modules of the simulation framework described in Chapter IV. In addition, the path following module is also executed, as well as the speech recognition engine (in asynchronous mode). When a new command is confirmed and processed, the system transitions to the **Command** state.
- d) The **Command** state is triggered when a user performs a command, it is confirmed and the target POI and command semantics are processed to update the vehicle path. The new path is then received and smoothed in this state. Afterwards, all the system states are initialized.
- e) Finally, the system reaches its **end** state when the vehicle arrives to its final destination.

Level 1: System Level States

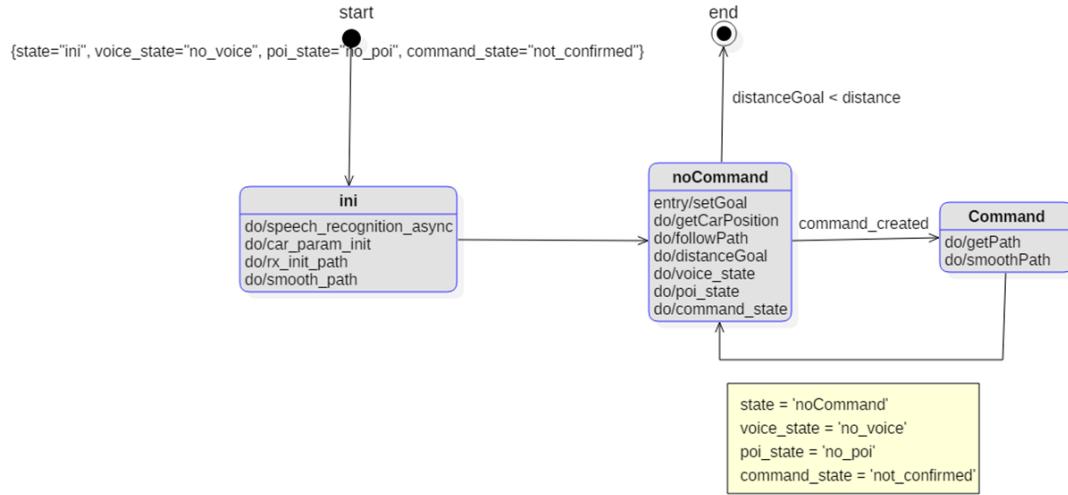


Figure 55. Level 1: System states.

2. Level 2: Voice Recognition states

Chapter VI described how the speech recognition engine allows to recognize the semantics of a voice-based user command. In order to keep track of the voice recognition process, this dissertation designed it as a two states state machine. In this way:

- a) **no_voice** represents the state in which no command semantics are recognized.
- b) When a user performs a voice command and its semantics are recognized by the speech recognition engine module, the voice system transitions to the **voice** state, which processes the semantics of the command and triggers the execution of the POI calculation state machine level.

As it was already mentioned, the objective of this state level is to process the command semantics. Thus, other interaction modes that allow to process them (for example, gesture control or brain signal recognition) could be used.

Level 2: Voice Recognition States

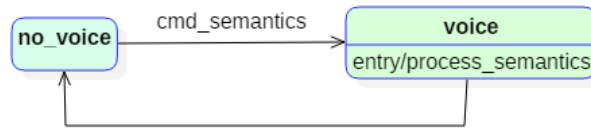


Figure 56. Level 2: Voice Recognition States.

3. Level 3: POI computation states

If the system recognizes the semantics of a voice command, and it did not received any command request yet (i.e. it is waiting for receiving a new command in the form described in Figure 5 of Chapter IV), the POI calculation state machine is triggered. This state machine controls the POI computation process.

- a) The **no_poi** state is in charge of executing the gesture recognition engine module. In this way, when the system receives the voice part of the command, the user is asked to point to the target POI. As it was described in Chapter IV, a real-world implementation of the system would consider the voice and pointing as parallel processes (similar to the work presented in [147], [148]). However, for simplification purposes, this step was performed sequentially in this dissertation.

- b) Once a user performs a pointing gesture the system transitions to the **waiting_poi_confirm** state. As the pointing gesture is subject to noise, more than a potential target POI can be detected, thus having POI ambiguity. In this state the system triggers the dialog feedback in order to solve this situation. If the system detects that only one POI is possible, it transitions to the **poi_confirmed** state. If the system detects that multiple POI are possible, it transitions to **waiting_poi_color** state. It is important to note that a more general development should be able to handle any POI properties related to the disambiguation process, and not only the color. Moreover, assuming that an annotated map is available, those characteristics should be retrieved from it. In this part of the dissertation, we did not assume any annotated map was available. Thus, the POI characteristics (color in this case) were retrieved using computer vision functions available in [123]. Other disambiguation use cases with annotated map information will be described latter in this chapter.
- c) As it was already mentioned, if more than one potential POI is detected the system transitions to **waiting_poi_color** state. By means of the dialog feedback system described in Figure 46 of Chapter VI as part of the ***Research Question 2***, the system and the user try to disambiguate the selection of target POI in the following way. When more than one target POI is detected, the system retrieves the colors corresponding to each potential target POI and the average coordinates corresponding to them. By means of the dialog feedback system described in Figure 46, the system asks to the user what building is he

/ she is referring to. In this way, the user can clarify to the system the color of the desired building. If this color corresponds to one of the potential building colors of the previous list, the system will retrieve its corresponding coordinates, construct the command and analyze its feasibility. At that point, if the command is feasible in terms of traffic rules compliance, it is assumed that the POI is confirmed and the system will directly transition to the **command_confirmed** state described latter in this chapter. If by any reason the user says a color that is not in the potential target POIs list, the system will inform the user about this, discard the command, and transition to the system initial states, ready to wait for new commands.

- d) If the system detects only one potential target POI, it will transition to the **poi_confirmed** state. In this state the system will calculate the POI coordinates, construct the command and analyze its feasibility. If the feasibility result is positive, the system will transition to the **waiting_confirmed** state. Otherwise, the system will be initialized to its initial states, ready to receive a new command.

Level 3: POI Computation States

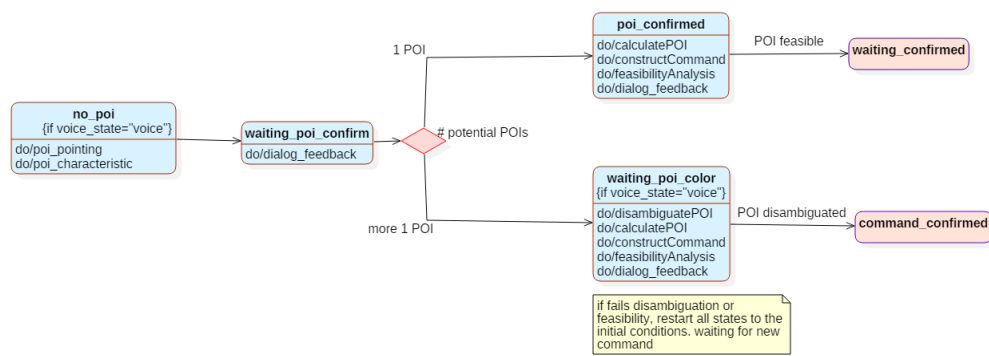


Figure 57. Level 3: POI computation states.

4. Level 4: Command States

When a voice and pointing gesture command is created, the 4th level of states is activated (Figure 58). This level controls the command status. In this way:

- a) **not_confirmed** state represents a state where no command structure (described in Chapter IV) was confirmed.
- b) When a user performs a command and the corresponding structure has been created, the system transitions to the **waiting_confirmed** state. This state is in charge of performing the dialog feedback that allows a user to confirm, reject or update a command, as it was described in Figure 48 of Chapter VI.

When a command is created, the dialog feedback system asks the user for confirmation. In this phase, the user can either confirm or reject the command.

If the user confirms it, the system will transfer to the **command_confirmed**

state. If the command understood by the system is not correct, the user have the following voice dialog (as shown in Figure 48):

- 1) Directly reject it. In this case, the system will initialize all its states and will be ready to accept new commands.
- 2) Reject it and update the POI properties. It may happen that the system miscalculated the POI, resulting in the wrong POI property (color in our case). The user can tell the system the correct POI property, so it can be updated and the new POI calculated.
- 3) Reject it and update the task. It may happen that the system misunderstand the intended task. In this case, in the same way as the previous case, the user may tell the system to correct it, so the task can be updated.
- 4) Reject it and update POI and task. It may happen that the system is wrong in POI property and task. In this case, the user is able to correct both, so the command can be properly updated.

After 2, 3 or 4 are performed, the system asks once more for confirmation. If the user confirms the new command, the system will transition to the **command_confirmed** state. Otherwise, the system will discard the command, initialize its states, and get ready to receive a new command.

In the case the user target POI color is no longer in the field of view of the vehicle, the system will give the proper feedback and discard the command.

This process is depicted in Figure 59.

- c) The **command_confirmed** state is triggered when a user confirms a command. This state is in charge of processing the command and transmit its corresponding fields to the vehicle path planning module. In this way, the path of the vehicle can be updated as described in Chapter VI. This state triggers the transition to the 1st level state **Command** previously described.

Level 4: Command States

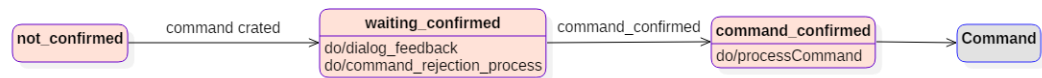


Figure 58. Level 4: Command States

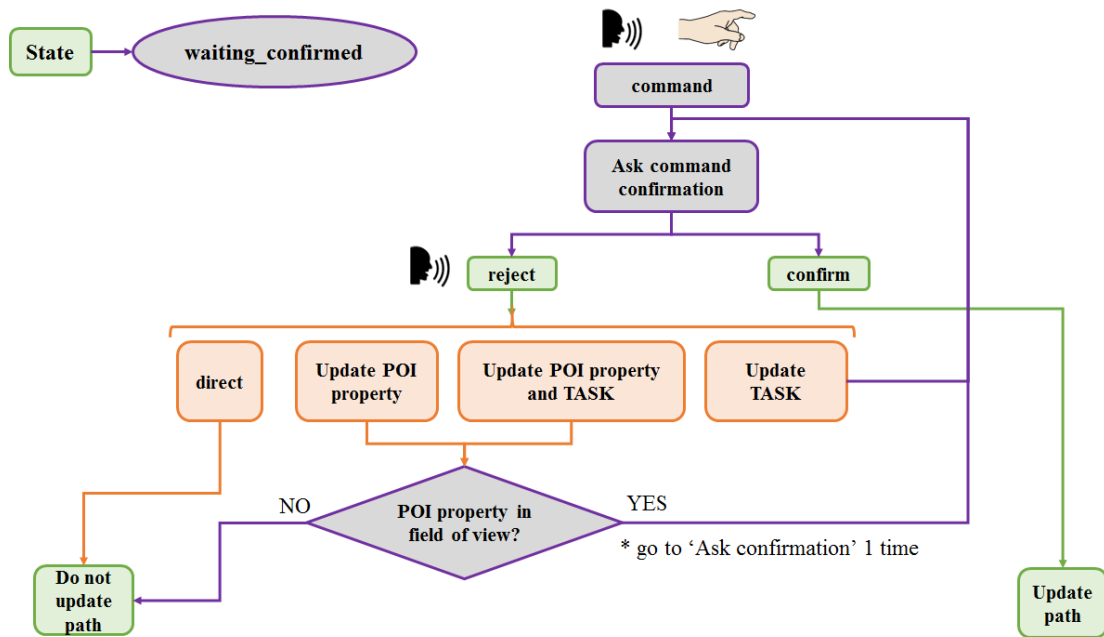


Figure 59. Command confirmation / rejection / update process.

This state design approach allowed a successful implementation of the system presented in this dissertation, reaching the experimental results described in Chapter VI.

C. Feasibility Analysis Module

When a user performs a voice and pointing gesture command, it will only be executed if it is feasible in terms of safety and traffic rules compliance. The feasibility analysis module is executed every time a new command is created. As it will be described in this subsection, to demonstrate this module a map database was created based on the 3D environment side of the simulation framework described in Chapter IV. In this way, certain coordinates were associated with a traffic element type. As it has been mentioned during this dissertation, if the user performs a command and the feasibility analysis module result is negative, the command will not be performed, the proper feedback will be given to the user and the vehicle will keep following its pre-defined autonomous path. Otherwise, the path planning module will update it. A simplified version of the overall process is depicted in Figure 60.

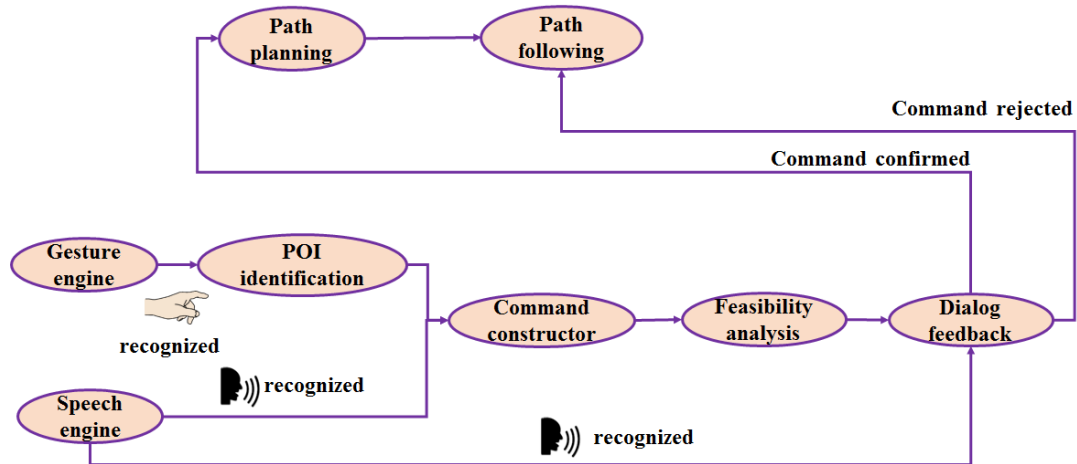


Figure 60. Feasibility analysis modules integrated in the system process.

A map database with three tables was created. The corresponding tables are presented in Figure 61.

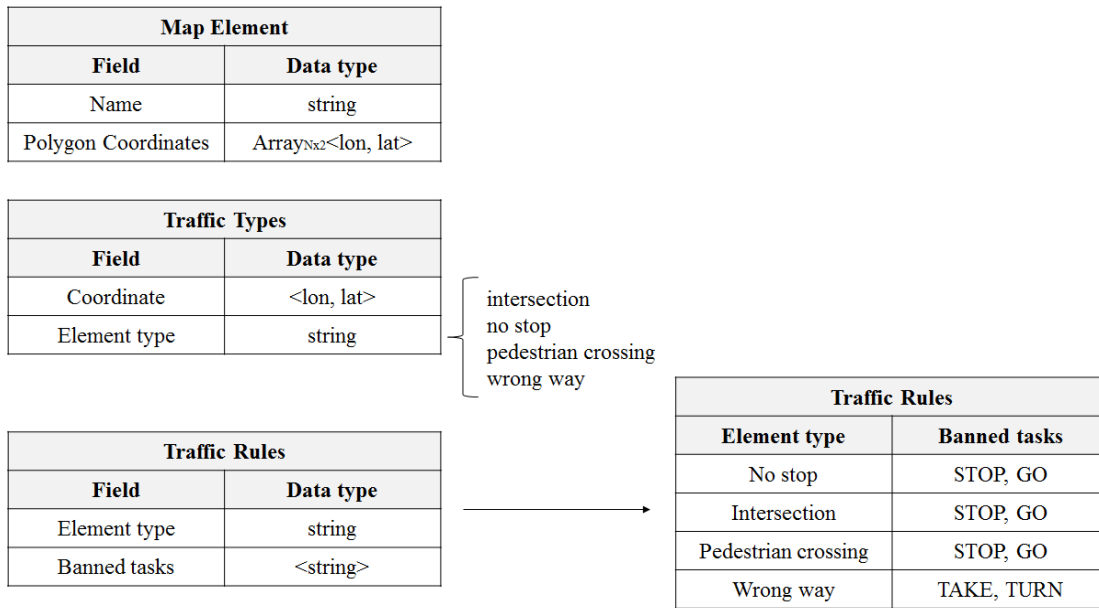


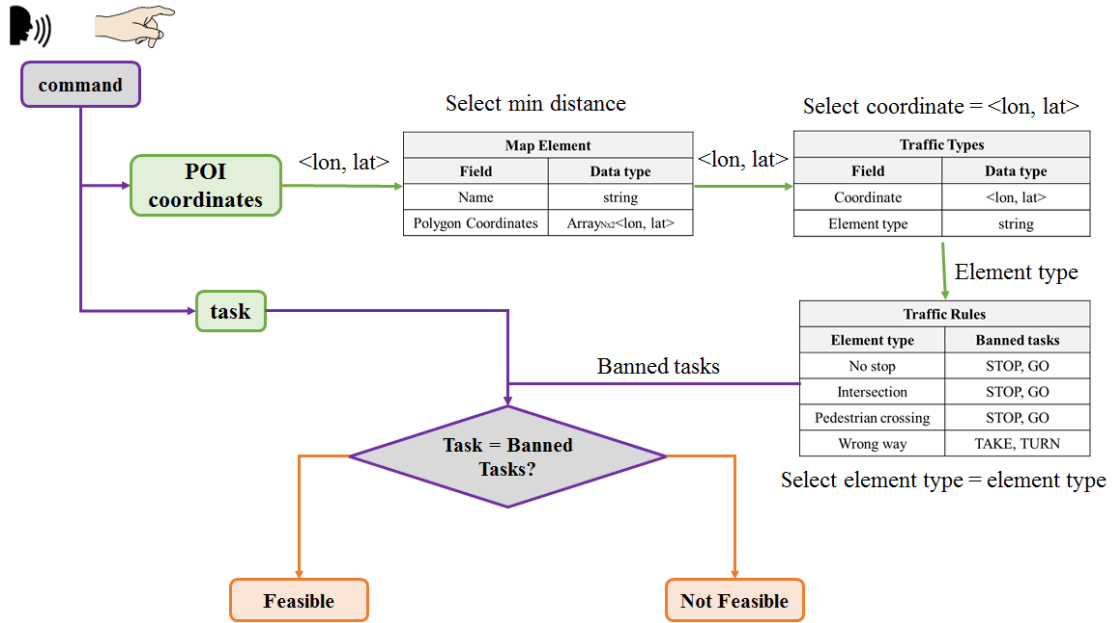
Figure 61. Map database tables.

The meaning of the tables in Figure 61 are the following:

- a) Map Element contains the mapped elements of our environment. Every map element has a name and an array of coordinates associated. For this particular dissertation the map elements are buildings and the associated coordinates form the perimeter of the building.
- b) Traffic Types associates coordinates with their corresponding element types. For example, in this dissertation four types were defined: intersection, no stop, pedestrian crossing, and wrong way.

- c) Traffic Rules associates an element type of the traffic type table with its corresponding banned task. For example, an intersection is an element where stopping is not allowed

Taking into account the map database that was just described, the feasibility analysis module defines the feasibility state of the performed command as it is depicted in Figure 62. When a user performs a command, it is processed by the feasibility analysis module. In this way, the coordinates of the command (target POI) are compared with the Map Element table in order to select the closest point in the database. The Traffic Types table determines if the selected point has an element type associated. The resulting element type is then searched in the Traffic Rules table in order to determine its corresponding banned tasks. If any of these banned tasks coincide with the command tasks, the command is considered not feasible and will not be executed. This feedback is given to the user, as it was shown in Figure 49.



D. POI disambiguation process for Research Question 1

The *Research Question 1* of this dissertation (Chapter V) analyzed the computation of the target POI coordinates. However, as it was shown the results were not 100% accurate. In *Research Question 2* the topic of POI disambiguation using a dialog feedback system based on voice was discussed (Chapter VI). In that case, the feedback was demonstrated in the simulation framework described in Chapter IV.

This subsection describes how two additional POI disambiguation use cases, applied to improve the POI computation accuracy of *Research Question 1*. As it was described in this chapter, this POI disambiguation would fit in the 3rd state level of Figure 54. In this particular case, for simplification purposes, voice feedback was not used, and the simplified feedback messages were prompted in the simulation command window.

1. POI disambiguation based on map database POI property

The method presented in the first part of *Research Question 1* was based on the comparison of the calculated target POI coordinates with an annotated map database. In this way, the selected map element was the one with minimum distance to the calculated target POI.

For this, a table of the minimum distances of the target POI to each of the elements of the map database was created, with their basic fields map element name and minimum distance coordinates. Using the method described in Chapter V the system will prompt a message with the detected target map element, and a menu with options to the user to specify if the selected is the correct map element or it is another one. If the user selects another map element, its name will be compared with the minimum distance table in order to extract the coordinates of the correct target POI. Figure 63 shows an overview of this process.

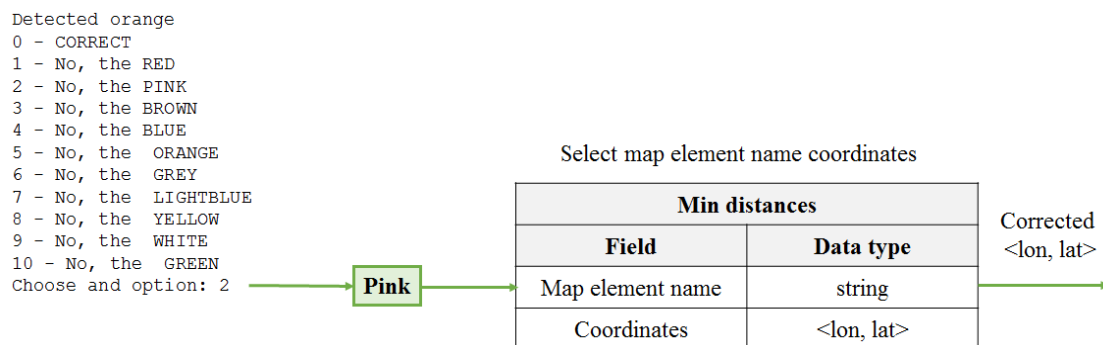


Figure 63. POI disambiguation process by map element name/property.

As it is shown, the system detects the orange building as the target POI. However, the user specifies that the correct one was the pink one. With this method, as it is depicted in Figure 64, the system can correct and update the target POI.

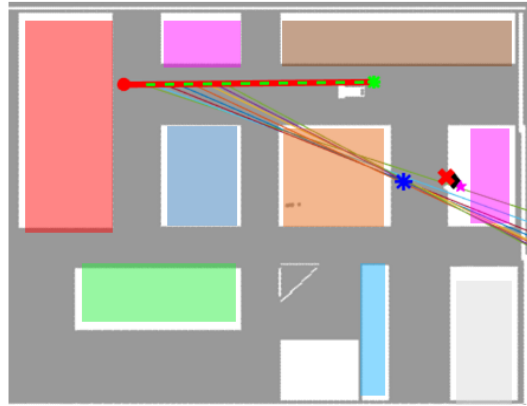
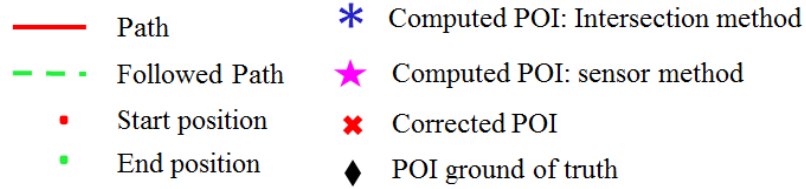


Figure 64. Result of POI disambiguation by map element name/ property.

As it is shown in Table 7 this method was able to correct all the POI inaccuracies of the simulated part of *Research Question 1*.

Table 7. POI disambiguation. Simulation accuracy results.

POI computation method		Intersection based	POI disambiguation improvement
Accuracy (%)	Non-cluttered	96%	(+4%) 100%
	Cluttered	76.5%	(+23.5%) 100%
	Total	86.25%	(+13.75) 100

2. POI disambiguation based on spatial information

Another method of POI disambiguation developed in this dissertation is based on user perceived spatial relation of map elements. In this way, given a map database with the fields of Table 8, map element name, coordinate of its center, and polygon coordinates (perimeter in this case).

Table 8. Map Elements database table.

Map Element	
Field	Data type
Name	string
Center Coordinate	<lon, lat>
Polygon Coordinates	Array _{Nx2} <lon, lat>

Applying the minimum distance method described in Chapter V, the system calculates a potential POI, prompts a message with the detected map element and a menu with options to the user to specify if the selected POI is the correct one or it is another one (Figure 65).

Detected Hangar 1
0 - CORRECT
1 - The building in FRONT
2 - The building BEHIND
3 - The building in the LEFT
4 - The building in the RIGHT
Select an option: 2

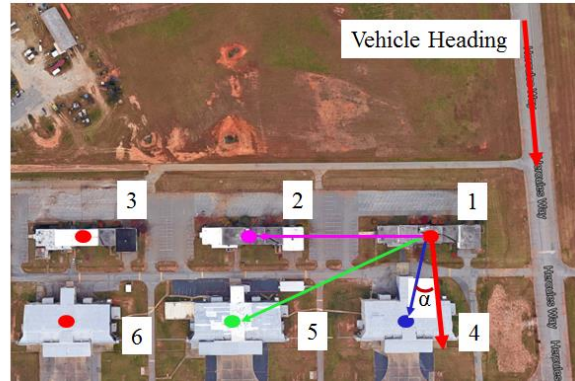


Figure 65. POI disambiguation process by spatial relation of map elements.

Considering the value of the angle α , it is possible to determine the spatial relations of the map elements with respect the pre-selected one, as it is depicted in Figure 66.

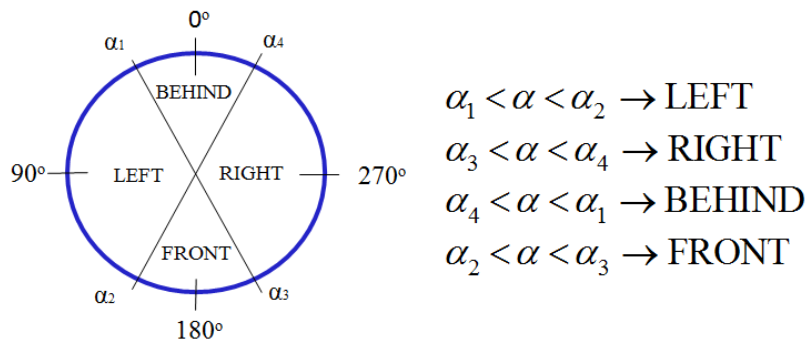


Figure 66. Map elements spatial relations based on alpha angle.

As it is shown, the system detects the Hangar 1 building as the pre-selected target POI. However, the user specifies that the correct one was the one behind it. With this method, as it is depicted in Figure 64, the system can correct and update the target POI.

- Computed POI: Intersection method
- ◆ Selected Map Element
- ✗ Corrected Map Element



Figure 67. Result of POI disambiguation by spatial relation.

As it is shown in Table 9 this method was able to correct all the POI inaccuracies of the real world experiments part of *Research Question 1*.

Table 9. POI disambiguation. Real world experiments accuracy results.

POI computation method		Intersection based	POI disambiguation improvement
Accuracy (%)	Cluttered	33.3%	(+66.7%) 100%

E. Summary

This chapter focused on answering the *Research Question 3* of this dissertation. The state machine design approach presented in this chapter allows to a successfully execution of the system presented in this dissertation. This approach enables the successful execution of the dialog feedback messages to the user (analyzed in Chapter VI), so that he / she can collaborate with the system to solve potential conflicting situations, such as POI ambiguity or feasibility of the intended command. In addition, this level structured state machine design enables a scalable and modular system in a way that it could allow to expand it by using other interaction modalities, as well as POI properties.

The method presented for the feasibility analysis of the user command is based on the comparison of the target POI with an annotated map that contains the traffic rules associated to certain coordinates of it. In this way, the traffic rules define what tasks are not allowed to be performed. If the task semantics of the user command are in this ‘not allowed’ set, the result of the feasibility analysis is negative and the command will not be executed (and the proper feedback message will be produced).

Another conflicting situation is related with POI ambiguity or error. In this chapter we explored how this ambiguity can be solved in the two scenarios that *Research Question 1* analyzed. In the simulation scenario a method based on POI property was proposed. On the other hand, in the real world scenario a method based on the target POI spatial relations was proposed. Both methods used the information available in an annotated map and were able to solve all the POI conflicting situations.

VIII. CHAPTER EIGHT: CONCLUSIONS AND FUTURE WORK

A. Conclusions

This dissertation presented a strategy for a multimodal human vehicle interaction system based on voice and pointing gesture command to enable users of autonomous cars to make and communicate situation awareness decisions to the vehicle, with the objective of commanding it towards a target point of interest; modifying, under feasible safe conditions, its pre-defined autonomous route. In this way, the interaction flexibility vs. user control/spontaneity dilemma that current autonomous vehicles approaches have can be mitigated.

In Chapter IV the system architecture that enabled this strategy was presented, as well as its basic functions, building blocks and command structure based on voice and pointing gestures. In addition, the simulation framework developed for this research was also described. This framework was based on a 3D virtual environment and a system processing environment. The 3D virtual environment provided the vehicle model and position, sensors (such as visual sensor), and path planning data, which was transmitted to the processing environment for performing the main functions of the system.

Chapter V concentrated on the POI computation challenge. First, a method for POI computation when the target POI is out of the sensors range was developed. This method was based on the intersections of pointing rays over the time of the pointing gesture process. In this way, these intersection points were used to calculate the coordinates of the target POI. This method is especially interesting for cluttered environments, where more than one target POI may intersect the pointing ray formed

during the pointing gesture process. The simulation results identified correctly the target POI 76.5% of the times in cluttered environment, while the traditional ray-casting method would have failed always.

The limited number of real world experiments (13 valid collected files: 7 for non-cluttered and 6 for cluttered environments) showed that the proposed method was 16.25% more accurate than the traditional ray-casting based POI computation method. However, due to current instrumentation limitations in the data collection experiments, this accuracy differs from the one resulting from the simulation results. Moreover, real world experiments may be subject to more sources of noise and inaccuracies.

The second part of this Chapter V presented a POI computation method for target POIs that are in the range of the sensors of the car. The simulation results showed that this method is able to identify the target POI correctly, with a distance error with respect to the ground of truth of $0.95 \pm 0.09\text{m}$ for non-cluttered conditions and $1.24 \pm 0.28\text{m}$ for cluttered ones.

In Chapter VI the role of voice was analyzed from two use cases. The first one, as a mean to extract the command semantics and combine them with the output of the POI computation in the same command structure, so that the system can update its pre-defined autonomous path depending on the meaning of those semantics. In this way, for example, the path update of the command semantics GO was treated in a different way than the command semantics TURN.

The second use case for the use of voice was related with the dialog feedback system as a method to solve in collaboration with the user potential conflicting situations, such as POI ambiguity or command not feasible in terms of traffic rules compliance.

The simulations in this part were performed using the namespaces of *System.Speech* [121]. The functions that this library contains allowed to perform speech recognition and synthesis processes needed for the speech recognition engine and dialog feedback module. Also, following the *W3C Recommendation “Speech Recognition Grammar Specification”* [122], custom built grammars were developed in order to recognize the semantics of the user voice commands, reaching an overall recognition accuracy of 79.5%.

Finally, in Chapter VII the system design approach followed for the development of the strategies and system presented in this dissertation was analyzed. A design approach based on level structured state machine allowed to create a modular system able to handle the conflicting conditions of POI ambiguity and command feasibility, as well as provide the proper feedback to the user at all times, and perform the user intended commands.

The method presented in this chapter for the feasibility analysis of the user command was based on the comparison of the target POI with an annotated map that contains the traffic rules associated to certain coordinates of it. In this way, when a user performed a command, the feasibility analysis module checked if the intended task was compliant with the traffic rules associated to the target POI.

Another conflicting situation was related with POI ambiguity or error. In this regards, using the information available in the annotated maps developed in this dissertation, two methods were explored to improve the POI computation results of Chapter V: one based on map element properties, such as color, and another one based on the spatial relation between the map elements. In both cases, all the POI identification errors were solved.

B. Future Work

This research work can be extended in different areas in order to make a more robust system.

The real world is more complex than the simulator framework presented in this dissertation. In addition, it is subject to more sources of error. Although in this dissertation the gesture recognition error was included in the simulated ‘click based pointing gesture’, other errors such as the external sensor errors or vehicle localization errors may be included in future implementations. It is considered also interesting to include other common modules of the autonomous vehicle architecture, such as perception, localization, collision avoidance, behavioral planning, or others. These were not included in this dissertation in order to focus on the key objectives of it. However, in a more realistic simulation scenario they would be needed.

As it was mentioned in Chapter V, a more refined instrumentation hardware would allow to better perform the pointing gesture experiments in a real world experimentation set up.

Although the speech recognition engine achieve accuracy rates of 79.5%, it has to be recognized that there is room for improvement in this regards. Especially when GARBAGE words are included in the speech. The modularity of the system presented in this dissertation would allow to test other speech recognition engines [152], [153].

As it was mentioned in Chapter VII, the level structured state machine system design approach would allow to explore the use of other interaction modes for retrieving the command semantics or calculating the target POI. Although in Chapter III of this dissertation voice combined with pointing gestures were identified as natural interactions modes for human-robot interaction, other interaction modes may be explored. Moreover, it could be interesting to even explore commands performed by people outside the car, similar to the work done in the field of human-robot interaction [96], [97].

It would be interesting to evaluate the proposed system in this dissertation from a user perspective. In this way, different experiments can be performed to study the potential differences in the way user perform pointing gestures or voice commands. For example, it would be interesting to explore the differences between left and right handed users, as well as users with different accents. In addition, user experiments may be performed in order to analyze the usability and user experience of the proposed system in this dissertation, in order to potentially adapt it to the output of those studies.

Finally, implementing parts of the system presented in this dissertation in a real car, with real sensor will give invaluable insight of other future challenges that may arise.

IX. APPENDIX: EXAMPLES OF SIMULATION OUTPUTS

This appendix provides some examples of the system execution.

A. Command not feasible in terms of traffic rules compliance

Figure 68 shows an example of a user performing a command that does not comply with the traffic rules. In this case, the system informs the user about this conflicting situation and asks for another command to try to solve it. As the second command is feasible, the system asks for command confirmation. Once it is confirmed, as it is seen in Figure 69, the new command is used to update the autonomous path.












 User	System 
 ok car stop there	Command semantics = OK CAR STOP THERE
	 Constructing command for: OK CAR STOP THERE. Click on your point of interest.
 * User points	<div>* Command not feasible</div>
	 I cannot perform STOP . It is not allowed at an intersection . Waiting for new command.
 ok car stop there	Command semantics = OK CAR STOP THERE
 * User points	<div>* Command feasible</div>
	 Did you mean STOP by the light blue building ?
 sure	Command semantics = CONFIRMED
	 Command confirmed Executing. Waiting for new command

Figure 68. Example of command not feasible.

Command
trigger: 'OK CAR'
task: 'STOP'
location: 'THERE'
object: ''
POI_x: 15.6719
POI_y: -13.3964
POI_color: 'lightblue'

- Original Path
 • Start position
 ◆ POI not feasible
- Updated Path
 * End position
 ◆ Final POI
- - - Followed Path
 * Original goal

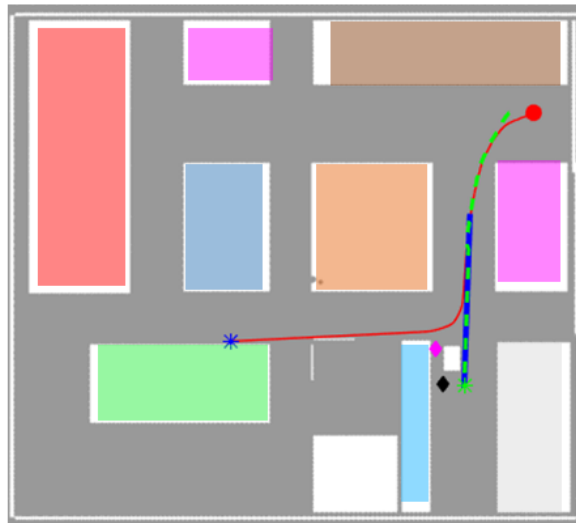


Figure 69. Resulting command and path for STOP command.

B. POI disambiguation

As it has been explained in this dissertation, it may happen that more than one potential target POI are detected by the system. In this case, as it is depicted in Figure 70,

by means of a user-system dialog, this ambiguity can be solved, and the new path created (Figure 71).

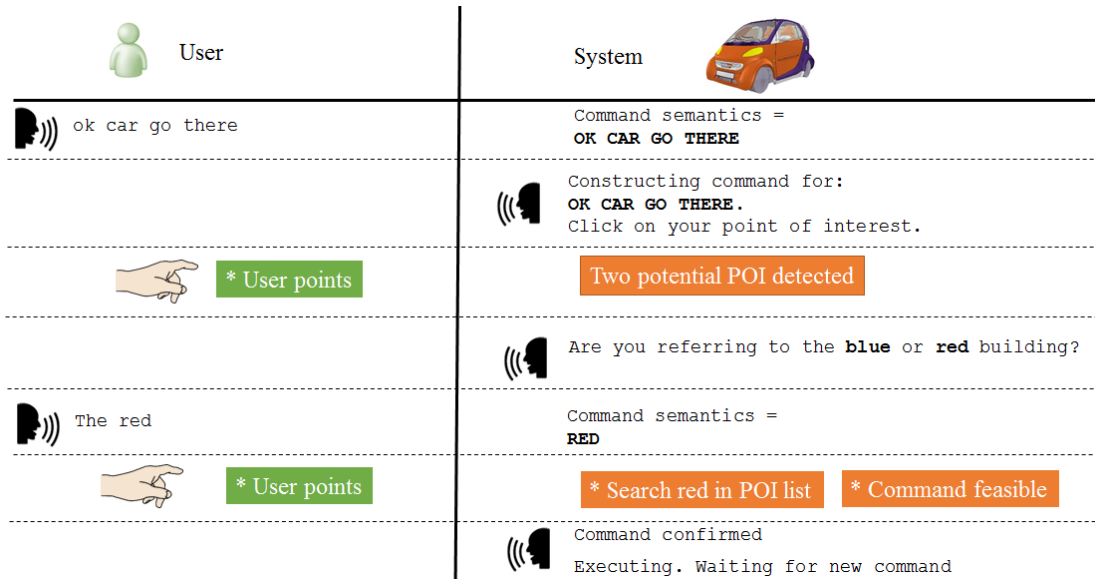


Figure 70. POI disambiguation dialog.

Command
trigger: 'OK CAR'
task: 'GO'
location: 'THERE'
object: ''
POI_x: -18.1774
POI_y: -3.4571
POI_color: 'RED'

- Original Path
 • Start position
- Updated Path
 * End position
 ◆ Final POI
- - Followed Path
 * Original goal

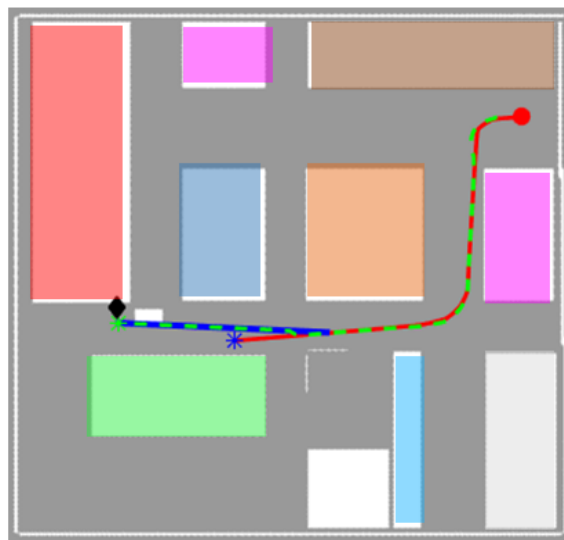


Figure 71. Result of POI disambiguation.

C. Correction of POI property: color

Figure 72 shows a dialog where the user corrects the POI property color. As it is shown in Figure 73, its new coordinates are computed and the new path is created (Figure 74).











 User	System 
 ok car stop there	Command semantics = OK CAR STOP THERE
	 Constructing command for: OK CAR STOP THERE. Click on your point of interest.
 * User points	* Ask for confirmation
	 Did you mean STOP by the blue building ?
 No, the green	Command semantics = NO COLOR:GREEN
	* Search green building centroid
	 Did you mean STOP by the green building ?
 Yes	Command semantics = CONFIRMED
	 Command confirmed Executing. Waiting for new command

Figure 72. Correction of POI color dialog.

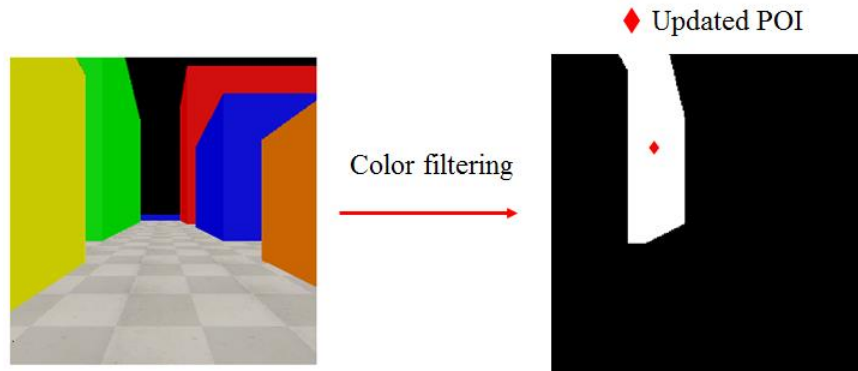


Figure 73. Color filtering to determine the new POI coordinates.

Command
trigger: 'OK CAR'
task: 'STOP'
location: 'THERE'
object: ''
POI_x: -4.7241
POI_y: -9.4115
POI_color: 'green'

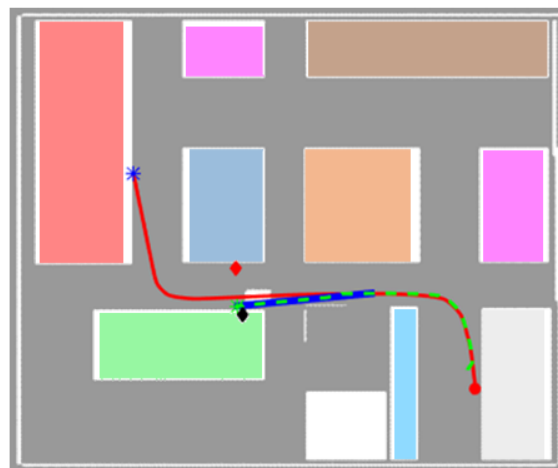
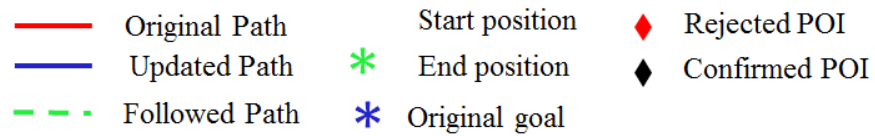


Figure 74. Result of POI color update.

D. Update command task and POI property (color)

Figure 75 shows a dialog where the user updates the command task and the POI color. In this case, as it is shown in Figure 75, the TAKE command sets the new POI as an intermediate landmark for the new path. It is also important to note that for this dissertation only POI and task were update, thus the semantics of location remain the same (however in further scenarios this could be updated as well).











 User	System 
 ok car stop there	Command semantics = OK CAR STOP THERE
	 Constructing command for: OK CAR STOP THERE. Click on your point of interest.
 * User points	* Ask for confirmation
	 Did you mean STOP by the pink building ?
 No, take the red	Command semantics = NO TASK:TAKE COLOR:RED
	* Search red building centroid
	 Did you mean TAKE the red building ?
 Yes	Command semantics = CONFIRMED
	 Command confirmed Executing. Waiting for new command

Figure 75. Dialog for task and color update.

Command
trigger: 'OK CAR'
task: 'TAKE'
location: 'THERE'
object: ''
POI_x: -17.6158
POI_y: 18.8423
POI_color: 'red'

- Original Path
 Start position
◆ Rejected POI
- Updated Path
 * End position
 ◆ Confirmed POI
- - - Followed Path
 * Original goal

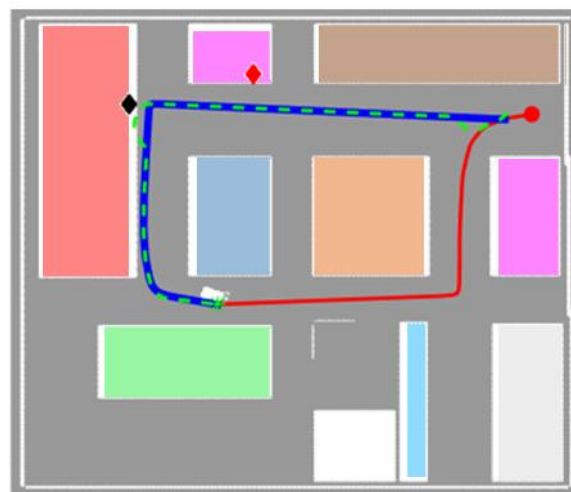


Figure 76. Result from task and color update.

X. LIST OF PUBLICATIONS

Published

1. **P. Sauras-Perez**, A. Gil-Batres, J. Singh Gill, P. Pisu, and J. Taiber, “VoGe: A voice and gesture system for interacting with autonomous cars”, *SAE Technical Paper* 2017-01-0068, 2017, doi:10.4271/2017-01-0068.
2. Jackeline Rios-Torres, **Pablo Sauras-Perez**, Ruben Alfaro, Joachim Taiber, Pierluigi Pisu, “Eco-Driver System for Energy Efficient Driving of an Electric Bus”, *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.* 8(1):2015.
3. Andrea Gil, **Pablo Sauras-Perez**, Joachim Taiber, “Communication Requirements for Dynamic Wireless Power Transfer for Battery Electric Vehicles”, *2014 IEEE International Electric Vehicle Conference (IEVC 2014)*.
4. **Pablo Sauras-Perez**, Joachim Taiber, John Smith, “Variability Analysis of In-Car Gesture Interaction”, *2014 International Conference on Connected Vehicles and Expo (ICCVE)*.
5. **Pablo Sauras-Perez**, Andrea Gil, Joachim Taiber, “ParkinGain: Toward a Smart Parking Application with Value-Added Service Integration”, *2014 International Conference on Connected Vehicles and Expo (ICCVE)*.
6. Jackeline Rios, **Pablo Sauras-Perez**, Andrea Gil, Andre Lorico, Joachim Taiber, Pierluigi Pisu., "Battery Electric Bus Simulator - A Tool for Energy Consumption Analysis," *SAE Technical Paper* 2014-01-2435, 2014, doi:10.4271/2014-01-2435.

To be submitted

7. **P. Sauras-Perez**, P. Pisu, and J. Taiber, “A voice and pointing gesture interaction system for on-route update of autonomous vehicle’s path”.
8. **P. Sauras-Perez**, P. Pisu, and J. Taiber, “Target Point of Interest computation in vehicle’s surroundings based on pointing rays intersections”.

Awards and presentations

VoGe: A voice and gesture system for interacting with autonomous cars

9. Top 7 in the 1st Valeo Innovation Challenge, 2014.
10. IEEE/ITIC Automotive Innovations Driving Experience. October, 2014.
11. Ingenious SC – 2015 SC Automotive Summit. February, 2015.
12. SAE Carolinas Meetup. March, 2017.

XI. REFERENCES

- [1] Continental Automotive – Advanced Driver Assistance Systems.
http://www.continental-automotive.com/www/automotive_de_en/themes/passenger_cars/chassis_safety/adas/. Last visited: March 2016.
- [2] E. Juliussen and J. Carlson, “Autonomous Cars – Not if, but when,” IHS Automotive, Jan. 2014.
- [3] Nissan’s Autonomous Self Driving Car | Nissan USA.
<http://www.nissanusa.com/blog/autonomous-drive-car>. Last visited: March 2016.
- [4] Model S Autopilot Press Kit | Tesla Motors.
<https://www.teslamotors.com/presskit/autopilot>. Last visited: March 2016.
- [5] “Autonomous Cars – An industry influence study,” Appinions Inc., July 2014.
- [6] “Self-Driving Cars: The Next Revolution,” KPMG, 2012.
- [7] “Self-Driving Cars: Are We Ready?” KPMG, 2013.
- [8] IEEE – News Releases, “Expert Members of IEEE Identify Driveless Cars as Most Viable Form of Intelligent Transportation, Dominating the Roadway by 2040 and Sparking Dramatic Changes In Vehicular Travel,” IEEE, Sept. 2012.
http://www.ieee.org/about/news/2012/5september_2_2012.html. Last visited: March 2016.
- [9] M. Montemerlo et. al., “Junior: The Stanford Entry in the Urban Challenge,” in *Journal of Field Robotics*, 2008.
- [10] Google Self-Driving Car Project. <https://www.google.com/selfdrivingcar/>. Last visited: March 2016.
- [11] IEEE – News Releases, “IEEE Survey Reveals Mass-Produced Cars Will Not Have Steering Wheels, Gas/Brakes Pedals, Horns, or Rearview Mirrors by 2035,” IEEE, July 2014. http://www.ieee.org/about/news/2014/14_july_2014.html. Last visited: March 2016.
- [12] Waymo. <https://waymo.com/>. Last visited: March 2017.
- [13] Unlock and authentication for autonomous vehicles, by D.T.Z Lu et. al., Google Inc. (2015, Nov. 24). U.S. Patent No. 9,194,168.

- [14] J. Markoff, "Google's Next Phase in Driverless Cars: No Steering Wheel or Brake Pedals," The New York Times, May 2014.
<http://www.nytimes.com/2014/05/28/technology/googles-next-phase-in-driverless-cars-no-brakes-or-steering-wheel.html>. Last visited: March 2016.
- [15] M. McFarland, "Look inside Google's new self-driving car," The Washington Post, July 2015.
<https://www.washingtonpost.com/news/innovations/wp/2015/07/13/look-inside-googles-new-self-driving-car/>. Last visited: March 2016.
- [16] K. Dagmar and A. Schmidt. "Design space for driver-based automotive user interfaces," In *Proceedings of the 1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. ACM, 2009.
- [17] C. Rodel et. al., "Towards Autonomous Cars: The Effect of Autonomy Levels on Acceptance and User Experience," In *Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. ACM, 2014.
- [18] L. De Ambroggi and A. Kona, "Google's driverless car to boost revenues for semiconductor," IHS Technology, June 2014.
<https://technology.ihs.com/502831/automotive-semiconductor-impression-220514>. Last visited: March 2016.
- [19] B. Schoettle and M. Sivak, "A survey of public opinion about autonomous and self-driving vehicles in the U.S., the U.K., and Australia", University of Michigan Transportation Research Institute UMTRI, July 2014.
- [20] J.D. Rupp and A. G. King, "Autonomous Driving – A Practical Roadmap," SAE International, 2010.
- [21] N. DeMattia, "Autonomous Driving: Coming in Small Doses," BMW Blog, 2015,
<http://www.bmwblog.com/2015/04/07/autonomous-driving-coming-in-small-doses/>. Last visited: March 2016.
- [22] A. Davies, "Ford's Skipping the Trickiest Thing About Self-Driving Cars," WIRED, Nov. 2015. <http://www.wired.com/2015/11/ford-self-driving-car-plan-google/>. Last visited: March 2016.
- [23] SAE On-Road Automated Vehicle Standards Committee. "Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems," SAE International, 2014.

- [24] B. W. Smith and J. Svensson, “Automated and Autonomous Driving: Regulation under Uncertainty,” Organization for Economic Co-operation and Development (OECD), 2015.
- [25] B. W. Smith, “Publications – NewlyPossible,” <https://newlypossible.org/wiki/index.php?title=Publications>. Last visited: March 2016.
- [26] B. W. Smith. "How Governments Can Promote Automated Driving." Available at Social Science Research Network (SSRN), 2016.
- [27] S.V. Casley et. al., “A study of public acceptance of autonomous cars,” Worcester Polytechnic Institute, Worcester, MA, USA, 2013.
- [28] D. J. Fagnant and K. Kockelman, “Preparing a Nation for Autonomous Vehicles: Opportunities, Barriers and Policy Recommendations for Capitalizing on Self-Driven Vehicles,” in *Transportation Research Part A*, July 2013.
- [29] N. Kalra, J. Anderson and M. Wachs, “Liability and Regulation of Autonomous Vehicle Technologies,” California PATH Program, Institute of Transportation Studies, University of California, Berkeley, April 2009.
- [30] T. Jiang et. al., “Self_Driving Cars: Disruptive or Incremental?” in *Applied Innovation Review*, June 2015.
- [31] California Department of Motor Vehicles, “Order to Adopt- Article 3.7 Autonomous vehicles,” August, 2014.
http://apps.dmv.ca.gov/about/lad/pdfs/auto_veh2/adopted_txt.pdf. Last visited: March 2016.
- [32] C. Urmson, “The View from the Front Seat of the Google Self-Driving Car, Chapter 3,” https://medium.com/@chris_urmson/the-view-from-the-front-seat-of-the-google-self-driving-car-chapter-3-476ea9deed9a#.ny2z0sdhw. Last visited: March 2016.
- [33] P. A. Hemmersbaugh, “Letter from NHTSA to Dr. Chris Urmson,” NHTSA, 2016.
<http://isearch.nhtsa.gov/files/Google%20-%20compiled%20response%20to%2012%20Nov%20%2015%20interp%20request%20--%204%20Feb%2016%20final.htm>. Last visited: March 2016.
- [34] “Federal Automated Vehicles Policy. Accelerating the Next Revolution in Road Safety.” USDOT, September 2016.

- [35] J. Youngs, “2014 U.S. Automotive Emerging Technologies Study Results,” J.D. Power, McGraw Hill Financial, 2014. <http://www.jdpower.com/cars/articles/jd-power-studies/2014-us-automotive-emerging-technologies-study-results>. Last visited: March 2016.
- [36] T. Adams, “Self-driving cars: from 2020 you will become a permanent backseat driver,” The Guardian, 2015. <http://www.theguardian.com/technology/2015/sep/13/self-driving-cars-bmw-google-2020-driving>. Last visited: March 2016.
- [37] AAA News Room. “Americans Feel Unsafe Sharing the Road with Fully Self-Driving Cars,” AAA, 2017. <http://newsroom.aaa.com/2017/03/americans-feel-unsafe-sharing-road-fully-self-driving-cars/>. Last visited: March 2017.
- [38] The Mercedes-Benz F 015 Luxury in Motion. <https://www.mercedes-benz.com/en/mercedes-benz/innovation/research-vehicle-f-015-luxury-in-motion/>. Last visited: March 2016.
- [39] Rinspeed XchangeE, <http://www.rinspeed.eu/concept-galery.php?cid=25>. Last visited: March 2016.
- [40] M. Sparkes, “Google forced to add steering wheel to driveless cars,” The Telegraph, Aug. 2014. <http://www.telegraph.co.uk/technology/google/11051009/Google-forced-to-add-steering-wheel-to-driverless-cars.html>. Last visited: March 2016.
- [41] S. Rumelin, C. Marouane, A. Butz, “Free-hand pointing for identification and interaction with distant objects,” in *Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, ACM, 2013.
- [42] K. Matsumura et. al., “Stick’N Conversation: Stick In-car Conversation into Places Using Multi Person Finger Pointing Gestures,” in *Adjunct Proceedings of the 7th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, ACM, 2015.
- [43] S. Kang et al. “Do you see what I see: towards a gaze-based surroundings query processing system,” in *Proceedings of the 7th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, ACM, 2015.
- [44] Y. H. Kim, and T. Misu, “Identification of the driver’s interest point using a head pose trajectory for situated dialog systems,” in *Proceedings of the 16th International Conference on Multimodal Interaction*, ACM, 2014.

- [45] T. Misu et al. "Situating language understanding for a spoken dialog system within vehicles," in *Computer Speech & Language*, 34(1), 2015, pp. 186-200.
- [46] National Highway Traffic Safety Administration. "Visual-manual NHTSA driver distraction guidelines for in-vehicle electronic devices." National Highway Traffic Safety Administration (NHTSA), Department of Transportation (DOT), 2012.
- [47] M. Geiger, et. al. "Intermodal differences in distraction effects while controlling automotive user interfaces," in *Proceedings Usability Evaluation and Interface Design, HCI*, 2001.
- [48] C. Müller and G. Weinberg. "Multimodal input in the car, today and tomorrow," *IEEE MultiMedia*, 2011.
- [49] C. Pickering, et. al. "A review of automotive human machine interface technologies and techniques to reduce driver distraction," in *2nd International Conference on Institution of Engineering and Technology (IET)*, 2007.
- [50] U. Reissner. "Gestures and Speech in Cars," in *Electronic Proceedings of Joint Advanced Student School (JASS)*, 2007.
- [51] B. Pfleging et. al. "Multimodal interaction in the car: combining speech and gestures on the steering wheel," in *Proceedings of the 4th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. ACM, 2012.
- [52] Gesture-based automotive controls, by N. Hobbs et. al., Google Inc. (2015, Jan. 27). U.S. Patent No. 8,942,881.
- [53] Interacting with vehicle controls through gesture recognition, by A. G. King et. al., Ford Global Technologies LLC. (2013, Aug. 8) U.S. Patent Application No. 13/366,388.
- [54] Interacting with a mobile device within a vehicle using gestures, by T. S. Paek et. al., Microsoft (2013, June 20). U.S. Patent: US 2013/0155237.
- [55] A. Riener et. al., "Standardization of the In-Car Gesture Interaction Space," in *Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, ACM, 2013.
- [56] BMW Technology Group: iDrive.
http://www.bmw.com/com/en/insights/technology/technology_guide/articles/idrive.html. Last visited: April 2016.
- [57] Android Auto. <https://www.android.com/auto/>. Last visited: April 2016.

- [58] CarPlay – Apple. <http://www.apple.com/ios/carplay/>. Last visited: April 2016.
- [59] Ford SYNC. <https://www.ford.com/technology/sync/>. Last visited: April 2016.
- [60] Chevrolet MyLink. <http://www.chevrolet.com/mylink-vehicle-technology.html>. Last visited: April 2016.
- [61] M. Alpern, and K. Minardo. “Developing a car gesture interface for use as a secondary task,” in *Extended abstracts on Human Factors in computing systems, CHI’03*, ACM, 2003.
- [62] A. Riener, and M. Rossbory. “Natural and Intuitive Hand Gestures: A Substitute for Traditional Vehicle Control?” in *Proceedings of the AutomotiveUI*, ACM, 2011.
- [63] A. Riener, et. al., “Natural DVI based on intuitive hand gestures,” *Workshop UX in Cars, Interact*, 2011.
- [64] M. Chiesa, “Experimenting Kinect interactions in the car,” in *Proceedings of the 4th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 2012.
- [65] E. Ohn-Bar et. al., “Hand gesture-based visual user interface for infotainment,” in *Proceedings of the 4th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, ACM, 2012.
- [66] C. Pickering, et. al., “A research study of hand gesture recognition technologies and applications for human vehicle interaction,” in *3rd Conference on Automotive Electronics*, 2007.
- [67] C. Endres, Christoph et. al. “Geremin: 2D microgestures for drivers based on electric field sensing,” in *Proceedings of the 16th International Conference on Intelligent User Interfaces*, ACM, 2011.
- [68] A. Riener, and P. Wintersberger. “3D Theremin: A Novel Approach for Convenient “Point and Click” Interactions in Vehicles,” in *3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI’11)*, ACM, 2011.
- [69] F. Carrino et. al. “In-vehicle natural interaction based on electromyography,” in *Proceedings of the 4th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, ACM, 2012.

- [70] L. Angelini et. al. "Opportunistic synergy: A classifier fusion engine for micro-gesture recognition," in *Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, ACM, 2013.
- [71] J. Suarez, and R. R. Murphy, "Hand gesture recognition with depth images: A review," in *IEEE RO-MAN*, IEEE, 2012.
- [72] P. Głomb, Przemysław, et al. "Choosing and modeling the hand gesture database for a natural user interface," in *Gesture and sign language in human-computer interaction and embodied communication*, Springer Berlin Heidelberg, 2011.
- [73] S. A. Grandhi, Sukeshini et. al. "Understanding naturalness and intuitiveness in gesture production: insights for touchless gestural interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2011.
- [74] J. C. Lee, "In search of a natural gesture," *ACM Crossroads*, 16.4 (2010): 9-12.
- [75] P. Sauras-Perez, Pablo et. al. "Variability analysis of in-car gesture interaction," in *2014 International Conference on Connected Vehicles and Expo (ICCVE)*. IEEE, 2014.
- [76] F. V. Zerlina et. al. "Gesture-based Interface Cognitive Work Evaluation in a Driving Context," IE486 Final Project, School of Industrial Engineering, Purdue University, 2010.
- [77] BMW AirTouch Press Release.
<https://www.press.bmwgroup.com/global/article/detail/T0247964EN/bmw-group-at-the-ces-2016-in-las-vegas-bmw-presents-the-principle-of-the-contactless-touchscreen-with>. Last visited: April 2016.
- [78] V. E. W. Lo, and P. A. Green, "Development and evaluation of automotive speech interfaces: useful information from the human factors and the related literature," in *International Journal of Vehicular Technology*, 2013.
- [79] U. Gärtner et. al. "Evaluation of manual vs. speech input when using a driver information system in real traffic," in *International Driving Symposium on Human Factors in Driving Assessment, Training and Vehicle Design*, 2001.
- [80] J. D. Lee, et al. "Speech-based interaction with in-vehicle computers: The effect of speech-based e-mail on drivers' attention to the roadway," in *Human Factors: The Journal of the Human Factors and Ergonomics Society* 43.4 (2001): 631-640.

- [81] M. Peissner et. al. "Can voice interaction help reducing the level of distraction and prevent accidents. Meta-Study on Driver Distraction and Voice Interaction," White Paper, Fraunhofer IAO and Carnegie Mellon University, 2011.
- [82] A. Baron, and P. Green, "Safety and usability of speech interfaces for in-vehicle tasks while driving: A brief literature review," No. UMTRI-2006-5. University of Michigan, Transportation Research Institute, 2006.
- [83] Z. Hua, and L. Ng. Wei, "Speech recognition interface design for in-vehicle system," in *Proceedings of the 2nd International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, ACM, 2010.
- [84] I. Alvarez, et al. "Voice interfaced vehicle user help," in *Proceedings of the 2nd International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, ACM, 2010.
- [85] I. Alvarez et al. "Designing driver-centric natural voice user interfaces," in *Adjunct Proceedings of AutomotiveUI*, ACM, 2011.
- [86] M. L. Seltzer, et al. "Robust location understanding in spoken dialog systems using intersections," in *INTERSPEECH*, 2007.
- [87] Mercedes-Benz History of Innovation | Mercedes-Benz.
<http://www.mbusa.com/mercedes/benz/innovation>. Last visited: April 2016.
- [88] Gesture actuated point of interest information systems and methods, by C. E. McCall et. al., Gm Global Technology Operations, Inc. (2010, Oct. 28). U.S. Patent Application No. 12/430,389. Application No. 12/430,389.
- [89] T. Grossman, and R. Balakrishnan, "The design and evaluation of selection techniques for 3D volumetric displays," in *Proceedings of the 19th annual ACM Symposium on User Interface Software and Technology*, ACM, 2006.
- [90] A. Steed, "Towards a general model for selection in virtual environments," in *IEEE Symposium on 3D User Interfaces*, IEEE, 2006.
- [91] Dang, Nguyen-Thong. "A survey and classification of 3D pointing techniques," in *2007 International Conference on Research, Innovation and Vision for the Future*, IEEE, 2007.
- [92] M. Wang, and D. Latotzky. "Driving an autonomous car with eye tracking," Free University of Berlin, 2010.

- [93] D. Göhring, et al. "Semi-autonomous car control using brain computer interfaces," in *Intelligent Autonomous Systems 12*, Springer Berlin Heidelberg, 2013, pp. 393-408.
- [94] J. Richarz, et al. "There you go!-estimating pointing gestures in monocular images for mobile robot instruction," in the *15th IEEE International Symposium on Robot and Human Interactive Communication*, IEEE, 2006.
- [95] D. Droeschel et. al. "Learning to interpret pointing gestures with a time-of-flight camera," in *Proceedings of the 6th International Conference on Human-Robot Interaction*, ACM, 2011.
- [96] H. Holzapfel, K. Nickel, and R. Stiefelhagen, "Implementation and Evaluation of a Constraint-Based Multimodal Fusion System for Speech and 3D Pointing Gestures." In *Proceedings of the 6th International Conference on Multimodal Interfaces*, ACM, 2004.
- [97] R. G. Boboc, et. al. "Point-and-command paradigm for interaction with assistive robots," in *International Journal of Advanced Robotic Systems 12*, 2015.
- [98] R. A. Bolt, "Put that there: Voice and Gesture at the Graphics Interface," in *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, 1980.
- [99] A. Kranstedt et al. "Deixis: How to determine demonstrated objects using a pointing cone," in *Gesture in Human-Computer Interaction and Simulation*. Springer Berlin Heidelberg, 2005, pp. 300-311.
- [100] G. Kowadlo, P. Ye, and I. Zukerman, "Influence of Gestural Saliency on the Interpretation of Spoken Requests," in *INTERSPEECH*. 2010.
- [101] L. She, et al. "Teaching robots new actions through natural language instructions," in *23rd IEEE International Symposium on Robot and Human Interactive Communication, 2014 RO-MAN*, IEEE, 2014.
- [102] T. Becker, et al. "Natural and intuitive multimodal dialogue for in-car applications: The SAMMIE system," in *Frontiers in Artificial Intelligence and Applications*, 2006.
- [103] S. Larsson, and J. Villing. "The DICO project: A multimodal menu-based in-vehicle dialogue system," In *Proceedings of the 7th International Workshop on Computational Semantics (IWCS-7)*, 2007.

- [104]J. Villing, et al. "Interruption, resumption and domain switching in in-vehicle dialogue," in *Advances in Natural Language Processing*. Springer Berlin Heidelberg, 2008. 488-499.
- [105]S. Kammel, et al. "Team AnnieWAY's autonomous system for the DARPA Urban Challenge 2007," in *Journal of Field Robotics* 25.9 (2008): 615-639.
- [106]N. E. Du Toit, et. al., "Situational Reasoning for Road Driving in an Urban Environment." in *Proceedings of the 2nd International Workshop on Intelligent Vehicle Control Systems (ICINCO 2008)*.
- [107]P. G. Trepagnier, et al. "Team Gray's 2007 Urban Challenge Vehicle," in *2007 DARPA Urban Challenge*, Technical Paper, 2007.
- [108]U. Ozguner et. al. "The Ohio State University Autonomous City Transport (OSU-ACT)," in *2007 DARPA Urban Challenge*, Technical Report, 2007.
- [109]S. Brechtel, et. al., "Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs," in *2014 IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2014.
- [110]V. Gadeppally, et al. "Driver/vehicle state estimation and detection," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2011.
- [111]A. Kurt, and Ü. Özgüner, "A probabilistic model of a set of driving decisions," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2011.
- [112]D. O. Sales, et al. "Adaptive finite state machine based visual autonomous navigation system," in *Engineering Applications of Artificial Intelligence* 29 (2014): 152-162.
- [113]D. O. Sales, et al. "FSM-based visual navigation for autonomous vehicles," in *Workshop on Visual Control of Mobile Robots-IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.
- [114]M. Buzdalov, and A. Sokolov, "Evolving EFSMs solving a path-planning problem by genetic programming," in *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*. ACM, 2012.
- [115]M. Amoozadeh, et al. "Platoon management with cooperative adaptive cruise control enabled by VANET," *Vehicular Communications* 2.2 (2015): 110-123.

- [116] T. J. Koo, et al. “Hierarchical approach for design of multi-vehicle multi-modal embedded software,” in *Embedded Software*, pp:344-360. Springer Berlin Heidelberg, 2001.
- [117] I. Traore, ed. “Continuous Authentication Using Biometrics: Data, Models, and Metrics,” IGI Global, 2011.
- [118] D. King-Hele, “Erasmus Darwin's improved design for steering carriages—and cars,” in *Notes and records of the Royal Society* 56, no. 1 (2002): 41-62.
- [119] The Open Motion Planning Library. <http://ompl.kavrakilab.org/>. Last visited: March 2017.
- [120] Coppelia Robotics V-REP. <http://www.coppeliarobotics.com/>. Last visited: March 2017.
- [121] System.Speech Namespaces. [https://msdn.microsoft.com/en-us/library/gg145021\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/gg145021(v=vs.110).aspx). Last visited: March 2017.
- [122] Speech Recognition Grammar Specification Version 1.0. <https://www.w3.org/TR/speech-grammar/>. Last visited: March 2017.
- [123] P.I. Corke, Robotics, Vision & Control: Fundamental Algorithms in MATLAB. Springer, 2011. ISBN 978-3-642-20143-1.
- [124] Y. Jiang, et al. “Target object identification and localization in mobile manipulations,” in *2011 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, 2011.
- [125] E. W. Weisstein, “Line-Line Intersection,” from MathWorld - A Wolfram Web Resource. <http://mathworld.wolfram.com/Line-LineIntersection.html>. Last visited: April 2016.
- [126] Kinect for Windows SDK. <https://msdn.microsoft.com/en-us/library/dn799271.aspx>. Last visited: April 2016.
- [127] USGlobalSat. <http://usglobalsat.com/>. Last visited: April 2016.
- [128] National Marine Electronics Association (NMEA). <http://www.nmea.org/>. Last visited: April 2016.

- [129] Programming Kinect for Windows v2.
<https://channel9.msdn.com/Series/Programming-Kinect-for-Windows-v2>. Last visited: April 2016.
- [130] Kinect – Windows app development. <https://dev.windows.com/en-us/kinect>. Last visited: March 2016.
- [131] <https://jonnyboats.wordpress.com/2009/06/>. Last visited: April 2016.
- [132] M. Tenney, “Microsoft Kinect – An Overview of Working with Data.” in *CAST Technical Publications Series. Number 10459*.
<http://gmvc.cast.uark.edu/uncategorized/working-with-data-from-the-kinect/>. Last visited: April 2016.
- [133] C. B. Park et. al., “Real-time 3D pointing gesture recognition in mobile space,” in *8th IEEE International Conference on Automatic Face & Gesture Recognition (FG’08)*. IEEE, 2008.
- [134] N. S. M. Nor, et al. “Tracking and detection of pointing gesture in 3D space,” in *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE, 2012.
- [135] C. B. Park, and L. Seong-Whan. “Real-time 3D pointing gesture recognition for mobile robots with cascade HMM and particle filter,” in *Journal of Image and Vision Computing 29.1* (2011): 51-63.
- [136] R. J. Schalkoff, “Pattern Recognition: Statistical, Syntactic and Neural Approaches,” *John Wiley and Sons*, 1992, ISBN 0-471-52974-5.
- [137] H. W. Sorenson, “Kalman filtering: theory and application,” IEEE, 1985.
- [138] J. Shu et. al. “Application of extended Kalman filter for improving the accuracy and smoothness of Kinect skeleton-joint estimates,” in *Journal of Engineering Mathematics 88.1* (2014): 161-175.
- [139] OpenGL. <https://www.opengl.org/>. Last visited: March 2017.
- [140] OpenGL Projection Matrix. http://www.songho.ca/opengl/gl_projectionmatrix.html. Last visited: March 2017.
- [141] N. Mirnig, et al. “A case study on the effect of feedback on itinerary requests in human-robot interaction,” in *20th IEEE international Symposium on Robot and Human Interactive Communication, RO-MAN*. IEEE, 2011.

- [142] EmguCV: OpenCV in .NET (C#, VB, C++ and more). <http://www.emgu.com/>. Last visited: March 2016.
- [143] OpenCV. <http://opencv.org/>. Last visited: March 2016.
- [144] Arduino – Home. <https://www.arduino.cc/>. Last visited: March 2016.
- [145] Minoru 3D. <http://www.minoru3d.com/>. Last visited: March 2016.
- [146] G. Bradski, and A. Kaehler, “Learning OpenCV: Computer vision with the OpenCV library,” O'Reilly Media, Inc., 2008.
- [147] Z. Prasov, and J. Y. Chai. “Fusing eye gaze with speech recognition hypotheses to resolve exophoric references in situated dialogue.” in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2010.
- [148] J. Y. Chai, P. Hong, and M. X. Zhou. “A probabilistic approach to reference resolution in multimodal user interfaces.” in *Proceedings of the 9th International Conference on Intelligent User Interfaces*. ACM, 2004.
- [149] S. Thrun. “Artificial Intelligence for Robotics,” Udacity Course. Online Learning. <https://www.udacity.com/>. Last visited: April 2016.
- [150] G. M. Hoffmann, et al. “Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing,” in *Proceedings of the 26th American Control Conference*. IEEE, 2007.
- [151] R. Solea, and U. Nunes. “Trajectory planning and sliding-mode control based trajectory-tracking for cybercars,” in *International Journal of Integrated Computer-Aided Engineering* 14.1 (2007): 33-47.
- [152] iOS – Siri – Apple. <http://www.apple.com/ios/siri/>. Last visited: March 2017.
- [153] Alexa. <https://developer.amazon.com/alexa>. Last visited: March 2017.