Blekinge Institute of Technology

Masters' Thesis

Computer Science

Thesis no: MCS-2010-32

Mobile Robot Navigation using Gaze Contingent Dynamic Interface

Authors:
Zaheer Ahmed
Aamir Shahzad

Supervisor:
Prof. Dr. Craig A. Lindley

September 28, 2010

This thesis is submitted to the School of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Computer Science. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Authors:
Zaheer Ahmed
Address: Folkparksvagen 16:05
Email: virtualtalks@gmail.com

Aamir Shahzad
Address: Folkparksvagen 16:05
Email: phulsphy@gmail.com

University Advisor(s):
Prof. Craig A. Lindley
School of Computing

School of Computing
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/com
Phone : +46 457 38 50 00
Fax : + 46 457 102 45

Contents

List of Figures

List of Tables

Abstract

Using eyes as an input modality for different control environments is a great area of interest for enhancing the bandwidth of human machine interaction and providing interaction functions when the use of hands is not possible. Interface design requirements in such implementations are quite different from conventional application areas. Both command-execution and feedback-observation tasks may be performed by human eyes simultaneously. In order to control the motion of a mobile robot by operator gaze interaction, gaze contingent regions in the operator interface are used to execute robot movement commands, with different screen areas controlling specific directions. Dwell time is one of the most established techniques to perform an eye-click analogous to a mouse click. But repeated dwell time while switching between gaze-contingent regions and feedback-regions decreases the performance of the application. We have developed a dynamic gaze-contingent interface in which we merge gaze-contingent regions with feedback-regions dynamically. This technique has two advantages: Firstly it improves the overall performance of the system by eliminating repeated dwell time. Secondly it reduces fatigue of the operator by providing a bigger area to fixate in. The operator can monitor feedback with more ease while sending commands at the same time.

Abbreviations

HCI	Human Computer Interaction
EOG	Electro-OculoGrahpy
POG	Photo-OculoGraphy
VOG	Video-OculoGraphy
MIS	Minimal Invasive Surgery
COM	Component Object Model
FM	Frequency Modulation
RC	Radio Control
IR	Infrared
PWM	Pulse Width Modulation
IPP	Integrated Performance Primitives
ATL	Active Template Library
SGCI	Static Gaze Contingent Interface
DGCI	Dynamic Gaze Contingent Interface
ANOVA	ANalysis Of VAriance

Chapter 1

Gaze Contingent Dynamic Interface for Tele-operation

In this chapter we present our research article published in STeP 2010: 14th Finish Artificial Intelligence conference, Espoo Finland 17-18 August 2010. Formal writing of the thesis is presented from chapter 2 onwards.

Gaze Contingent Adaptive Interface for Tele-operation

Zaheer Ahmed[⋆†]
[⋆]Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden
virtualtalks@gmail.com

Aamir Shahzad[†]
[†]Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden
phulsphy@gmail.com

Craig A. Lindley[‡]
[‡]Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden
craig.lindley@bth.se

Abstract

Using eyes as an input modality for different control environments is a great area of interest for enhancing the bandwidth of human machine interaction and provide interaction functions when the use of hands is not possible. Interface design requirements in such implementations are quite different from conventional application areas. Both command-execution and feedback-observation tasks are performed by human eyes simultaneously. In order to control the motion of a mobile robot by operator gaze interaction, gaze contingent regions in the operator interface are used to execute robot movement commands, with different screen areas controlling specific directions. Dwell time is one of the most established techniques to perform an eye-click analogous to a mouse click. But repeated dwell time while switching between gaze-contingent regions and feedback-regions decreases the performance of the application. We have developed a dynamic gaze-contingent interface in which we merge gaze-contingent regions with feedback-regions dynamically. This technique has two advantages: Firstly it improves the overall performance of the system by eliminating repeated dwell time. Secondly it reduces fatigue of the operator by providing a bigger area to fixate in. The operator can monitor feedback with more ease while sending commands at the same time.

1 Introduction

In Human Computer Interaction (HCI) it is critical to study both the context of task performance and the role of attention, including visual focus of attention. A user's gaze is a strong indicator of intention or attention (Zhai, 2003). Moreover, when functioning well, the eyes are the primary source of information about the context and details in the environment as a basis for action (Kumar et al., 2008). Hence using the eyes as an input modality compared to more conventional input modalities (key-board, mouse, joy-stick etc.) has been an area of great interest in the field of HCI (Levine, 1981; Bolt, 1982; Ware and Mikaelian, 1987; Robert J.K., 1990). Factors that give inspiration and motivation for this include (Zhai et al., 1999):

1. In some of situations both of a user's hands are constantly engaged with other tasks, and disabled users may not be able to use their limbs.
2. Eye movement is faster than the other parts of the body. The process of acquiring and activating a target point on an interface using a cursor involves two steps: visually focusing the target first and then performing the actuation of the cursor. This means that if we can track the eye gaze successfully and use it accurately and efficiently, no other input source can be as fast as eye.
3. Key-board and pointing devices may be a cause of exhaustion and potential damage. This is another factor of concern in the field of HCI. Eye gaze as an input modality may be a nice solution to these problems.

As eye-tracking systems become more accessible, there are an increasing number of demonstrations of the use of eye-tracking to control the motion of controllable physical agents like robots (Bolt, 1982). Eye-tracking can provide an accurate gaze point on an object on a computer screen that the user is looking at (e.g. Tobii T60). In robot control systems, eye gaze direction can be used as an input source, similar to conventional input modalities: mouse, key-board, joy-stick etc. As such, gaze-based interaction has been used extensively to provide interaction capability for computer users who lack adequate use of their hands and arms, e.g. for word processing, writing email, and using the web (SmartboxAssistiveTechnology, 2010). Gaze contingent interfaces for robots can provide further assistance to those with disabilities, in the form of systems for manipulation (e.g. by directing robot arms) or for exploration (e.g. controlling the movement of mobile

robots). Eye-tracking is also being explored as a method for enhancing human-robot communication, e.g. to allow humanoid robots to react to human eye movements during conversational interactions (Atienza and Zelinsky, 2002). As an example of gaze contingent interfaces for the disabled, (Lin et al., 2006) describes an interface developed to control a powered wheel-chair. This interface is operated by human eye gaze. The interface is divided into 9 regions. Of those 9 regions, 4 are gaze contingent or command regions and remaining 5 regions are idle regions. Stop command is sent to the wheel-chair when user's gaze falls into these regions. Gaze contingent regions in an eye gaze controlled interfaces are those regions that are used to trigger specific commands when the eye gaze falls within them. In (Lin et al., 2006), screen regions are used to initiate wheel-chair motion commands. Similar work is presented in (Barea et al., 2002), where electrooculography (detection of electrical impulses from muscles that control eye gaze direction) is used to send driving commands to a wheel-chair from the interface. In both of these works, the operator is sitting on the wheel-chair. No feedback is provided to the operator through the interface, i.e. the interface is used only for one-way communication.

An experimental eye-tracking algorithm has also been used to control a robotic arm (Yoo et al., 2002). For this experiment, the interface is divided into 2 regions: a command region and the feedback region. Feedback is provided in the form of images taken by a camera installed in the robot location. Similar work has been presented for control of a robotic toy (Bhuiyan et al., 2004). In this work a different eye-tracking technique is used: in order to find out the gaze direction, location of the eye ball in the eye socket is tracked. In this technique the only purpose of the interface is to present feedback to the operator. In the more general field of robotics (i.e. beyond concerns with human disability) most research addresses controllable agents rather than fully autonomous agents (Olsen and Goodrich, 2003). In most cases these controllable agents are required to be controlled from a remote location, an approach called tele-operation (Latif et al., 2008). TeleGaze (Latif et al., 2008) and Gaze-controlled Driving (Tall et al., 2009) are recent projects in the area of robotics and control systems using eye gaze contingent interfaces for more natural human robot interaction. In TeleGaze, the interface is divided into 9 gaze contingent regions for different commands. These gaze contingent regions take 1/3 second dwell time to activate and issue the associated command to a WIFI enabled modified wheel-chair robot. Dwell time enables eye gaze to act as mouse click. When the operator fixates in a gaze contingent region for a predetermined interval of time (e.g. 1/3 sec), it is considered as an activation similar to a mouse click and a command is issued to the system to be controlled.

Tele-operation is characterized by controlling some system from a remote location (Latif et al., 2008). It involves two separate actions: one is to monitor the current state of the system to be controlled and the second is to send the appropriate commands according to the current state of the system. Monitoring is performed by the human eye and the hands are responsible for command execution through conventional input modalities like a key-board, mouse or joy-stick. If eye gaze alone is used as an input modality for tele-operation, it is obvious that both monitoring and command execution must be performed by eye gaze. The eye can fixate (i.e. focus at a point on the screen to keep a cursor stationary at a gaze contingent region for a certain interval of time) (Lankford, 2000) only on one object at a time; i.e. when the operator tries to focus on feedback (Monitoring) she loses focus on command execution interface, and vice versa (this applies in all of (Lin et al., 2006), (Barea et al., 2002), (Latif et al., 2008) and (Lankford, 2000). The resulting constant switching between two focus areas decreases the performance of the system considerably. The main reason for this performance deterioration is dwell time. Dwell time reinitializes to zero if the operator loses focus within a gaze contingent region in the interface. So switching between feedback-monitoring and command-execution takes repeated dwell time to send commands, resulting in performance deterioration.

An alternative design, described below, is that if an operator fixates in a gaze sensitive region and after the dwell time whole feedback region becomes the command execution region as well, then the overheads of switching between command execution and feedback regions is eliminated. This is explained in detail in the next section.

2 Interface Design

We seek an answer to the following research question in this paper: How to avoid multiple dwell time and improve performance of the overall activity of tele-operation? In order to answer this question we decided to analyze gaze behavior of the participants on a similar kind of interface proposed in related

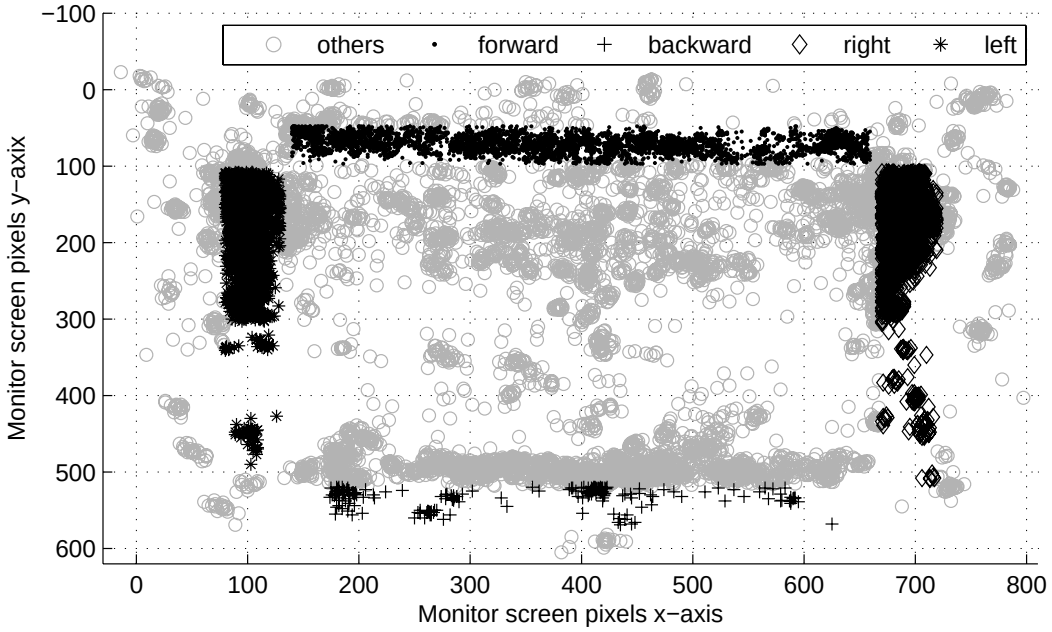

Figure 1: Gaze behaviour of the user.

work studies (Latif et al., 2008), in tasks controlling a wheeled mobile robot with visual feedback provided by a real time video link from a camera mounted on the robot platform. Gaze-directed control areas of the screen are mapped over the user's view of the video transmission from the robot. The robot can be driven in forward and backward directions, or turned by control of its differential drive system. Figure-1 is a scatter plot of a user's gaze fixations in different parts of the screen while interacting with an interface using fixed areas for dwell-activated command initiation. The participant users have used all the gaze contingent regions according to their needs. An interesting observation is that participants have tended to prefer the upper half areas of the RIGHT and LEFT regions. These regions are represented by $\diamond$ and $*$ in the plot, respectively. We used this observation in our alternative interface design and shortened these regions by 70 pixels from below, using this region instead for the STOP command in our alternative interface (Figure-2). Gray portions in the plot represent all those fixations when the participant focuses in the feedback region or some other regions in the monitor screen than gaze contingent command regions. In the new design, the STOP region dynamically adjusts its position in the interface. Initially it rests on top of the BACKWARD region. If a participant fixates in the BACKWARD region it changes its position and sticks to the FORWARD region at the top of interface. Whenever participant user switches between FORWARD and BACKWARD command regions, STOP region moves in between, since it is logical to stop before moving forward or backward. Our interface consists of 10 gaze contingent regions in total: 2 regions for STOP, 4 regions for FORWARD, BACKWARD, RIGHT and LEFT. The remaining 4 are dynamically formed by expansion of FORWARD, BACKWARD, RIGHT and LEFT. All other regions except STOP expand over the whole feedback region after the dwell time when fixated by the user. This provides more ease to the participant user since it is easy to fixate in comparatively larger regions. It also eliminates repeated dwell time for multiple commands. When one task is completed, the user fixates in some other gaze contingent region to perform a task associated to that region. As a result, the previous gaze contingent region shrinks back to its default location and the new region as selected by the user merges with the feedback region, and so on. This is a more appropriate technique for tasks that need multiple commands (e.g. turning to a suitable angle towards the right in multiple increments of 15°).

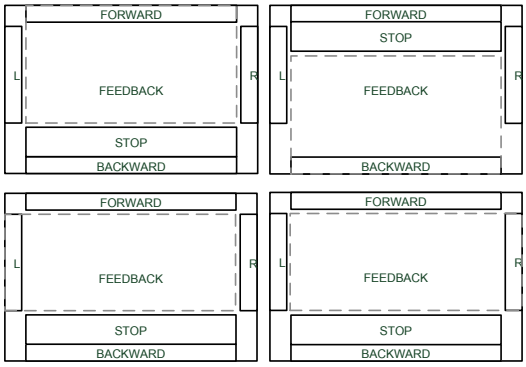

Figure 2: Four different states of the interface.

3 Method

Our empirical study is based on two experiments. Both the experiments are task-oriented. Human robot interaction applications are very diverse. Hence, there are no standard metrics for evaluation of newly developed applications (Latif et al., 2008). However, there are some common metrics in any application domain that are most likely to be used to evaluate the application developed in that particular domain (TobiiTechnologiesAB, 2010). Time needed to complete a task is a common measure (Tsui et al., 2009). Experiment 1 is a rather small experiment. The task in this experiment is to turn the robot at an angle of 90°. The reason for this separate experiment is that the robot we are using can turn only with a limited angle of 15° per turn command. In order to get larger turn angles, multiple turn commands are needed; e.g. with a command turn angle of 15° the robot needs 6 turn commands to turn a total angle of 90°. Forward and backward movements are far simpler than turning, since forward and backward motion is continuous. That is why we decided to examine the turning motion in a separate experiment. In experiment 2, a track is designed for the robot to travel on. An experiment participant interacts with the robot using gaze contingent interfaces and navigates through this track. Participants perform both of the experiments multiple times. Data is collected by noting the task completion time for each experiment trial.

3.1 Experiment Design

In our empirical study the independent variable is screen adaptiveness, which has two possible values: *static* and *adaptive*. These interface variants are used to control the navigation of a mobile robot. In both experiments participants complete a navigation and movement task. The outcome/dependent measure of the experiments is the task completion time. The upper half of Figure-3 shows the experiment 1 setup. A card of gray color is placed in front of the robot. Another card of black color is placed on the left side of the robot at an angle of 90° from the gray card. When a participant fixates in the gaze contingent region specified for LEFT, the robot starts turning to the left. When the turning robot completes an angle of 90° it comes in front of the black card. Now the participant can see the black card in the feedback region of the interface and should use gaze in the stop region of the interface to stop further movement. The participant performs this activity with both static and adaptive versions of the interfaces. In experiment 2, participants are given a task to send commands to the robot and monitor visual feedback, to drive the robot around a specified track to return to the starting location. The lower half of Figure-3 presents the layout of the track used in experiment 2.

We observed that the majority of the participants were not familiar with the concept and technology of eye tracking. This was a potential validity threat and could have effected the outcome due to varying capabilities and understanding of the participants. To bring them at an equal level of understanding we arranged training sessions for the participants. Another validity threat was power supplies to the logic and actuation parts of the mobile robot. Weak batteries may result in slow movement and delayed responses. To avoid this threat all the batteries were replaced with a fresh and fully charged set of batteries before switching to the next participant. XBee© transceivers are used for communication between application program and mobile robot, having a range of 120 meters without obstacles. We ensured that the mobile robot remained within this range while performing tasks. The track is designed keeping in mind that it

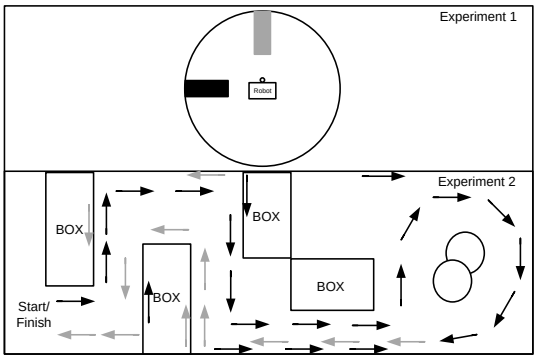

Figure 3: Task layout for experiment 1 and 2

should include all possible navigational moves. The total length of the track is 19.25 meters. The start and finish point are same. The whole track is marked with black and gray arrows. The participant follows black arrows to move down the track to the turning area and gray arrows for coming back to the finish point. 15 participants in total take part in these experiments. In experiment 1, 10 trials are recorded for each participant. This results in 15 x 10 = 150 trials of data. Experiment 2 is a bit lengthier in terms of time. Hence a block of 3 trials is recorded for each participant, giving 15 x 3 = 45 trials in total.

3.2 Apparatus and Conditions

An Intel© Core 2™ Extreme Q6850 @3.00GHz was programmed using Microsoft© Visual C++™ to create the gaze-directed monitoring and control interface. A Tobii© Eye Tracker T60™ was used to get participant's gaze behaviour and this was integrated with the interaction system via an API in order to use gaze data on the monitor screen. GrabBee© video grabber was used to capture the video stream from the wireless camera mounted on the head of the mobile robot. Intel's© image processing library OpenCV© was used to grab the video stream and then superimpose interface components on it. The Arduino© open-source electronics prototyping platform was used to configure and map robot actuation via different commands. XBee© serial communication transceivers were used to communicate commands between the user application software and the mobile robot. The interfaces developed in the project were flexible to work on any resolution but all experiments were conducted with 800x600 screen resolution.

3.3 Procedure

When participants arrived they were briefed about the experiments so that they could have an idea about the whole activity to be performed. Participants were seated 60-65 cm viewing distance from the eye-tracking device as directed in the Tobii© T60 user's manual. A calibration process was performed for each participant to get accurate gaze fixation points for each participant on eye-tracking screen. Then eye tracking was started. Each participant was asked whether the cursor is moving to the desired location of the screen with her gaze fixation or not. When the participant was satisfied with the result, the robot camera view with the interface superimposed upon it was presented to the participant. This concludes the experiment setup and now the participant can fixate in the gaze contingent regions to send commands to the robot. Each participant was allowed 3-4 training sessions to gain familiarity with the environment. Outcomes for actual trials were then recorded according to the pre-planned requirements.

3.4 Participants

Fifteen participants took part in this experiment. All were from Blekinge Institute of Technology (BTH). Out of fifteen, 10 were male and 5 female. Two faculty members also took part in this experiment. The age range of the participants was 20-51 years. In order to ensure that all the participants had the same level of understanding they were briefed in the same way about the experiment. They were also given a training session before data collection was started.

4 Results

Our claim is that mean task completion time for the dynamic interface design is less than for the static interface design. For this claim we have following null hypothesis (Ho) and alternate hypothesis (Ha).

Ho: Mean task completion time is same for both dynamic and static interfaces.
Ha: The dynamic interface takes less time to complete a task compared to the static interface.

In both the experiments the sample sizes are greater than 30, which is large enough to substitute sample variance with population variance. So we use a z-test (α = 0.05) for statistical analysis. Table-1 and Table-2 show the results. In our results, for experiment 1 the average task completion time for the dynamic interface is 74.62% of the static interface time. In this way the dynamic interface improves the performance of the system significantly, i.e. by 25.37% of the average static interface task completion time. For experiment 2, the average task completion time for the dynamic interface is 74.67% of the static interface time, so the dynamic interface improves the performance of the system by 25.32% of the average static interface task completion time. This improvement is due to the merger of the feedback region and the gaze-contingent region in the dynamic interface, which eleminates multiple dwell times that occur in the static interface design. In both experiments, z-stat is less than z-critical. So we can say that the difference between the populations is statistically significant. Hence the null

hypothesis can be rejected in favour of the claim that dynamic interface takes less time to complete a task as compared to the static interface.

Table-1: z-test results for experiment-1

Description	*Dynamic*	*Static*
Mean	2.966666667	3.976666667
Known Variance	0.061163	0.567707
Observations	150	150
Hypothesized Mean Difference	0	
z	-15.59863388	
P(Z<=z) one-tail	0.000000000	
z Critical one-tail	1.644853627	
P(Z<=z) two-tail	0.000000000	
z Critical two-tail	1.959963985	

Table-2: z-test results for experiment-2

Description	*Dynamic*	*Static*
Mean	115.2	153.6888889
Known Variance	2810.618182	3026.491919
Observations	45	45
Hypothesized Mean Difference	0	
z	-3.379424033	
P(Z<=z) one-tail	0.000363189	
z Critical one-tail	1.644853627	
P(Z<=z) two-tail	0.000726379	
z Critical two-tail	1.959963985	

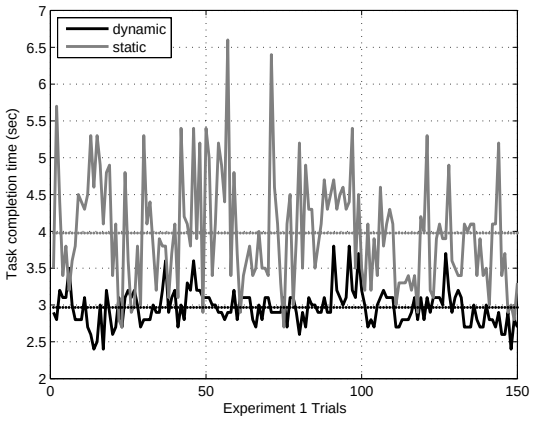

Figure 4: Experiment-1 results.

5 Conclusion and Discussion

In this paper we have introduced the concept of a gaze contingent dynamic interface to control a mobile robot. The basic purpose was to decrease the repeated dwell time of gaze directed robot control resulting from switching between feedback and gaze contingent regions. A second purpose of the study was to facilitate the operator in such a way that she feels more comfortable with feedback monitoring tasks while sending navigation commands to the robot at

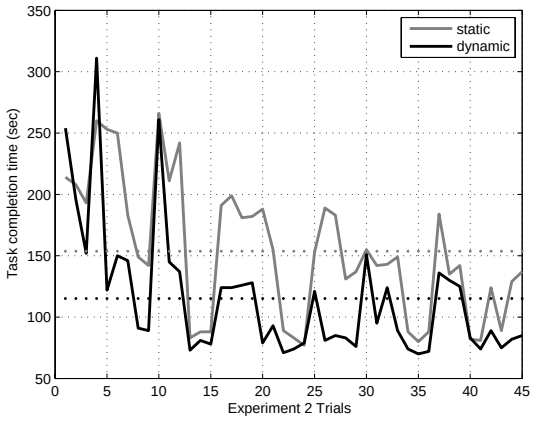

Figure 5: Experiment-2 results.

the same time. Results of our pilot study show that the performance of our proposed dynamic interface is significantly better than the static interface. Twelve out of fifteen participants reported that they felt more comfortable with using the dynamic interface. The rest of the participants voted in favour of the static interface. An interesting observation in this regard is that the participants who voted in favour of the static interface performed almost same with both interfaces i.e. their task completion time was almost same for both interfaces. All those participants who can drive or play computer games completed experiment task in less time while avoiding collisions, compared to the rest of the participants.

In almost all trials each participant performed quite well in every subsequent trail i.e. she took less task completion time.

Another interesting observation can be made regarding training of the participants. In initial trails we handled each participant alone at the experiment site and she learned from her own experience. But later in the experiments we worked with groups of three or four participants on the experiment site at the same time. In this scenario when each participant was performing the experiment, the remaining were watching her activities. All such participants who watched others, performed exceptionally well on their turn and produced less task completion times. This phenomenon can be seen in graphical results of experiment 2 (Figure-5). It is evident from the graph that task completion times for the initial 12 trials are very high and then we can see more consistent results for rest of the trials. This later region after 12 trials is the region where participants were present in groups.

In the current scenario each gaze contingent region

sends a single command to the robot. There is very little variation available in different activities. For example we can move left or right with a fixed angle or forward or backwards with a single speed. The dynamic interface has a quite big area. In a future variant of the interface this area could be used to bring variation in the degree of motion specified in commands. For example, the upper half area of the LEFT gaze contingent region can be used to turn at different angles. or the FORWARD region could be used to move with different speed levels. This may be explored in future work.

References

R. Atienza and A. Zelinsky. Active gaze tracking for human-robot interaction. *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces,IEEE Computer Society*, 2002.

R. Barea, L. Boquete, M. Mazo, and E. Lopez. System for assisted mobility using eye movements based on electrooculography. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10:209–218, 2002.

M. A. Bhuiyan, V. Ampornaramveth, S. Muto, and H. Ueno. On tracking of eye for human-robot interface. *International Journal of Robotics and Automation*, 19: 42–54, 2004.

R. A. Bolt. Eyes at the interface. *Human Factors in Computer Systems Gaithersburg,Maryland:ACM*, 1982.

M. Kumar, J. Klingner, R. Puranik, T. Winograd, and A. Paepcke. Improving the accuracy of gaze input for interaction. *Proceedings of the 2008 symposium on Eye tracking research applications Savannah, Georgia:ACM*, pages 65–68, 2008.

C. Lankford. Effective eye-gaze input into windows. *Proceedings of the 2000 symposium on Eye tracking research applications Palm Beach Gardens, Florida,United States: ACM*, pages 23–27, 2000.

H. Latif, N. Sherkat, and A. Lotfi. Telegaze: Teleoperation through eye gaze. *Cybernetic Intelligent Systems,2008. CIS 2008.7th IEEE International Conference*, pages 1–6, 2008.

J. L. Levine. An eye-controlled computer. *IBM TJ Watson ResearchCenter: Yorktown Heights,New York*, 1981.

Chern-Sheng Lin, Chien-Wa Ho, Wen-Chen Chen, Chuang-Chien Chiu, and Mau-Shiun Yeh. Powered wheelchair controlled by eye-tracking system. *Optica Applicata*, 36:401–12, 2006.

Dan R. Olsen and Michael A. Goodrich. Metrics for evaluating humanrobot interactions. *Performance Metrics for intelligent Systems (PerMIS 2003)*, 2003.

Jacob Robert J.K. You look at is what you get: Eye movement-based interaction techniques. *CHZ 90: ACM Conference on Human Factors in Computing Systems: Addison-Wesley/ACM Press*, 1990.

SmartboxAssistiveTechnology. *Software*, page [Online]. Available:http://www.smartboxat.com/software.html, 2010.

M. Tall, Hansen, A. Alapetite, D. W. Hansen, J. S. Agustin, E. Mllenbach, and H. H. Skovsgaard. Gazecontrolled driving. *27th International Conference Extended Abstracts on Human Factors in Computing Systems CHI 2009, April 4, 2009 - April 9, 2009, Boston, MA, United states: Association for Computing Machinery*, pages 4387–4392, 2009.

TobiiTechnologiesAB. *Tobii T60 and T120 Eye Trackers*, page [Online]. Available:http://www.tobii.com/scientific research/products services/eye tracking hardware/tobii t60 t120 eye trackers.aspx, 2010.

K. M. Tsui, D. J. Feil-Seifer, M. J. Matari, and H. A. Yanco. Performance evaluation methods for assistive robotic technology. *Performance Evaluation and Benchmarking of Intelligent Systems*, 2009.

C. Ware and H. H. Mikaelian. An evaluation of an eye tracker as a device for computer input. *CHI+GI:ACM Conference on Human Factors in Computing Systems and Graphics Interace.Toronto*, 1987.

Dong Hyun Yoo, Jae Heon Kim, KimDo Hyung, and Myung Jin Chung. A human-robot interface using vision-based eye gaze estimation system. *IROS 2002: IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 196–201, 2002.

S. Zhai. What is in the eyes for attentive input. *Communications of the ACM*, 46(3), 2003.

S. Zhai, C. Morimoto, and S. Ihde. Manual and gaze input cascaded (magic) pointing. *Proceedings of the SIGCHI conference on Human factors in computing systems:the CHI is the limit Pittsburgh, Pennsylvania,United States: ACM*, pages 246–253, 1999.

Chapter 2

Introduction

In Human Computer Interaction (HCI) it is critical to study both the context of task performance and the role of attention, including visual focus of attention. A user's gaze is a strong indicator of intention or attention [48]. Moreover, when functioning well, the eyes are the primary source of information about the context and details in the environment as a basis for action [20]. Hence using the eyes as an input modality compared to more conventional input modalities (keyboard, mouse, joy stick etc.) has been an area of great interest in the field of HCI [25, 7, 42, 33]. Factors that give inspiration and motivation for this include [49]:

1. In some situations both of a user's hands are constantly engaged with other tasks, and disabled users may not be able to use their limbs.

2. Eye movement is faster than the other parts of the body. The process of acquiring and activating a target point on an interface using a cursor involves two steps: visually fixating the target first and then moving the cursor to the activation point. This means that if we can track the eye gaze successfully and use it accurately and efficiently, no other input source can be as fast as eye.

3. Key board and pointing devices may be a cause of exhaustion and potential damage. This is another factor of concern in the field of HCI. Eye gaze as an input modality may be a nice solution to these problems.

As eye-tracking systems become more accessible, there are an increasing number of demonstrations of the use of eye-tracking to control the motion of controllable physical agents (robots) [7]. Eye-tracking can provide an accurate gaze point on an object on a computer screen that the user is looking at (e.g. Tobii T60 [20]). In robot control systems, eye gaze direction can be used as an input source, similar to conventional input modalities: mouse, key board, joy stick etc. As such, gaze-based interaction has been used extensively to provide interaction capability for computer users who lack adequate use of their hands and arms, e.g. for word processing, writing email, and using the web [35]. Gaze contingent interfaces for robots can provide further assistance to those with disabilities, in the form of systems for manipulation (e.g. by directing robot arms) or for exploration (e.g. controlling the movement of mobile robots). Eye-tracking is also being explored as a method for enhancing human-robot communication, e.g. to allow humanoid robots to react to human eye movements during conversational interactions [3].

2.1 Technical Terminologies and Definitions

In this section, basic terminologies are defined which will help the reader to understand the forth coming discussion about the core topic.

2.1.1 Eye Tracking

Eye Tracking is to track the eye(s) movement which also refers to locating the point where we look or fixate (i.e. focusing at a point on the screen) [21], that is called point of gaze. It is the process by which one can locate or measure the gaze point of the eye relative to the head [11].

Eye movement measurement methodologies can be divided into four broad categories: Electro-OculoGrahpy (EOG), Scleral Contact Lens/Search Coil, Photo-OculoGraphy (POG) or Video-OculoGraphy (VOG) and video-based combined pupil and corneal reflection [11]. The basic principle of Electro-Oculo-Graphy is dc-signal recordings of the electric potential differences of the skin surrounding the ocular cavity. The Scleral Contact Lens/Search Coil

method provides the most accurate eye movement measures. In this method a mechanical or optical reference object is attached to a contact lens and then this lens is worn by the test subject directly on the eye. POG or VOG involve analysis of the measurement of distinguishable features of the eyes under rotation/translation for example, the apparent shape of the pupil, the position of the limbus (the iris/sclera boundary) and corneal reflections of closely situated directed light source (often infra-red).

A problem with all these techniques is that they usually do not provide the point of regard measurement i.e. what the person is looking at.

This is specialty of the video-based trackers that they use image processing to calculate the point of regard in real time. It is observed in practice that the reflection from the cornea remains roughly constant in position during eye movement. This means that a reflection in the eye will remain in a static position during rotational movement of the eye and changes in gaze direction. This can be used to obtain a basic eye and head position reference. This retinal reflection provides a simple reference point to compare with the moving pupil. This comparison is used to calculate a gaze direction vector of the eye. The *Tobii T90* is a video-based eye tracker that we will use in our experiments [11].

2.1.2 Eye Gaze as an Input Modality

It is known that humans look at real world things and hold their eye gaze relatively still for a short interval of time, enough for the brain to perceive the nature of the features and form etc. of an object. These periods when the eyes rest upon a visual feature are referred to as *fixations* [33]. Fixations can be variably defined in terms of an angle within which the gaze point remains (e.g. 5°) and a period for which it must remain within this angle; a fixation point occurs when the eye remains within the specified angle for no less than the minimum specified period. The period is typically about 200-600ms. Fixations ensure that visual features of most interest fall within the foveal region of the retina (a high acuity field of vision). Everything out side this field is seen indistinctly. Between these fixations, gaze jumps rapidly

from one object to another. Such a movement is called a *saccade*. Saccades typically last for 30-120ms. There are also small scale eye movements that occur within fixations (tremor, drift, and microsaccades), but consideration of these is beyond the scope of this thesis.

Since the foveal field of visual acuity is fairly small (about 1 mm in diameter), and people need to direct their gaze vector exactly to the object of interest to get its accurate view. Because of this fact it is possible to trace the exact point of gaze from an eye fixation than then use it as an input modality.

2.1.3 Interface

The best definition of the interface, which also frees the term from that limited by the concepts of today, is that "interface" means exactly what the word roots connote: inter (between) and face, or that stuff that goes between the faces (i.e., senses) of the human and the machine [5].

2.1.4 Tele-operation

Most controllable agents need to be controlled from a remote location, which is commonly known as tele-operation [24].

2.1.5 Mobile Robot

"A mobile robot is a combination of various physical (hardware) and computational (software) components. In terms of hardware components, a mobile robot can be considered as a collection of subsystems for Locomotion: how the robot moves through its environment; Sensing: how the robot measures properties of itself and its environment; Reasoning: how the robot maps these measurements into actions; and Communication: how the robot communicates with an outside operator [12]."

Chapter 3

Background and Problem Definition

As an example of gaze contingent interfaces for the disabled, [26] describes an interface (Figure 3.1) developed to control a powered wheelchair. This interface is operated by human eye gaze. The interface is divided into 9 regions. Of those 9 regions, 4 are gaze contingent command regions. Gaze contingent regions in eye gaze controlled interfaces are those regions that are used to trigger specific commands when the eye gaze falls within them. In [26], screen regions are used to initiate wheelchair motion commands. Similar work is presented in [4], where electrooculography (detection of electrical impulses from muscles that control eye gaze direction) is used to send driving commands to a wheelchair from the interface. In both of these works, the operator is sitting on the wheelchair. No feedback is provided to the operator through the interface, i.e. the interface is used only for one-way communication.

An experimental eye-tracking algorithm has also been used to control a robotic arm [47] (Figure 3.2). For this experiment, the interface is divided into 2 regions: a command region and the feedback region. Feedback is provided in the form of images taken by a camera installed in the robot location. Similar work has been presented for control of a robotic toy [6]. In this work a different eye-tracking technique is used: in order to find out the

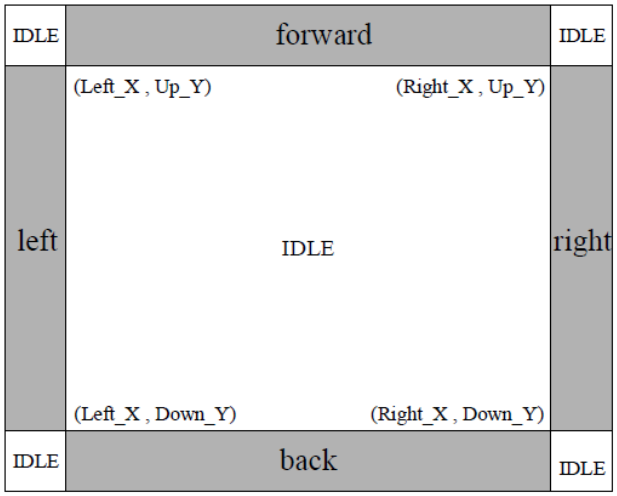

Figure 3.1: *Interface design for powered wheelchair [26]*

gaze direction, the location of the eye ball in the eye socket is tracked. In this technique the only purpose of the interface is to present feedback to the operator.

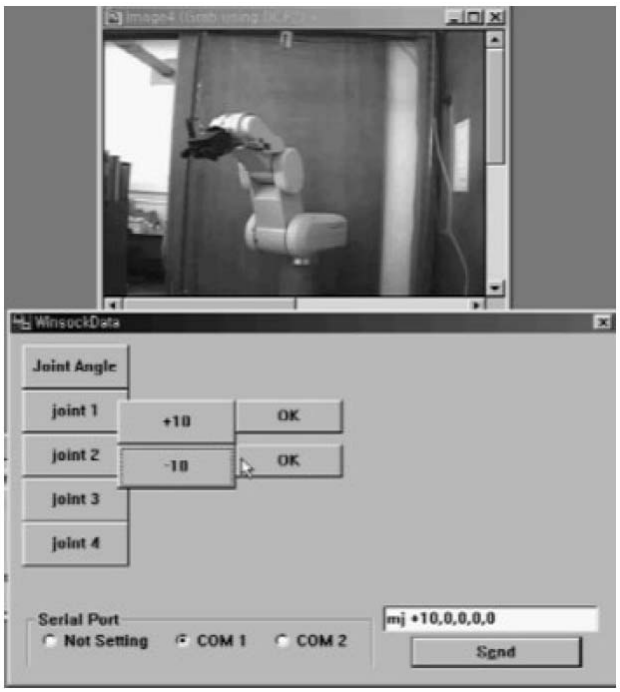

Figure 3.2: *Interface to control robotic arm [47]*

In the more general field of robotics (i.e. beyond concerns with human disability) most research addresses controllable agents rather than fully autonomous agents [31]. In most cases these controllable agents are required to be controlled from a remote location, an approach called tele-operation [23]. TeleGaze [23] and Gaze-controlled Driving [36] are recent projects in the area of robotics and control systems using eye gaze contingent interfaces for more natural human robot interaction. In TeleGaze, the interface (Figure 3.3 is divided into 9 gaze contingent regions for different commands. These gaze contingent regions take 1/3 second dwell time to activate and issue the associated command to a WIFI enabled modified wheelchair robot. Dwell time enables eye gaze to act as mouse click. When the operator fixates in a gaze contingent region for a predetermined interval of time (e.g. 1/3 sec), it is considered as an activation similar to a mouse click and a command is issued to the system to be controlled.

Figure 3.3: *Interface to control WIFI enabled powered wheelchair robot (TeleGaze) [23]*

3.1 Problem Definition

Tele-operation is characterized by controlling some system from a remote location [23]. It involves two separate actions: one is to monitor the current

state of the system to be controlled and the second is to send the appropriate commands according to the current state of the system. Monitoring is performed by the human eye and the hands are responsible for command execution through conventional input modalities like a key board, mouse or joy stick. If eye gaze alone is used as an input modality for tele-operation, it is obvious that both monitoring and command execution must be performed by eye gaze. The eye can fixate (i.e. focus at a point on the screen to keep a cursor stationary at a gaze contingent region for a certain interval of time) [22] only on one object at a time; i.e. when the operator tries to focus on feedback (Monitoring) she loses focus on command execution interface, and vice versa, if these display area are separate (this applies in all of [26], [4], [23] and [22]). The resulting constant switching between two focus areas decreases the performance of the system considerably. The main reason for this performance deterioration is dwell time. Dwell time reinitializes to zero if the operator loses focus within a gaze contingent region in the interface. So switching between feedback-monitoring and command-execution takes repeated dwell times to send commands, resulting in performance deterioration.

3.2 Challenge or Problem Focus

This thesis targets this performance problem for a gaze-directed robot control system inspired by TeleGaze [23]. We will try to investigate different possibilities for improving the interaction performance by making and evaluating different arrangements of the active regions in the interface. We will compare the results of this interface with conventional input modalities like key-board, mouse and joy-stick. We will also try to diagnose the limits of a TeleGaze-style system and reasons behind these limitations.

3.3 Aims and Objectives

Main goal with this project is to develop a gaze-contingent interface for mobile robot control inspired by the TeleGaze environment by integrating

different components of the environment: Eye Gaze Tracker, teleoperation station and a mobile robot. Using this environment we will perform several experimental trials in order to:

- Evaluate alternative interface designs with the goal of optimizing performance compared with other modes of interaction: key-board, mouse and joy-stick.
- Study the limits of the gaze-directed robot control and the reasons behind the limitations.

This study, we expect, will develop our understanding of Eye Gaze Tracking Technologies and Human Robot Interaction.

3.4 Research Questions

Question 1: How can different components, an Eye Gaze Tracker, a Teleoperation Station and a Mobile Robot, be integrated with each other to work as a whole system?

Question 2: How can we improve the performance of gaze interaction by making different arrangements of the active regions of the operator interface?

Question 3: What are the limitations of gaze interaction, if any, and the reasons for these limitations?

3.5 Expected Outcomes

- Description of Eye Gaze Tracking Technologies and how they work.
- Specification of experimental requirements for the Spinosaurus mobile robot platform (see appendix A for details).

- Specification, design and implementation of the experimental environment i.e. how different components of the environment are integrated with each other and their working principles.
- Evaluation results of different experiments.
- Details of the limitations, if any, of gaze-directed motion control for mobile robots and the reasons for these limitations.

Chapter 4

Method

In this chapter, we present the research methods used to achieve the goals of our study as described in previous chapter. The work reported in the chapters from 5 - 7 is done within the paradigm of constructive and empirical research. *Constructive* and *quantitative research* methods are used to conduct the study. The outcome of the constructive research is a "construct" which is a proposed solution and then this construct is evaluated using a prototype against predefined criteria.

4.1 Constructive Research

The constructive research approach is intended to find the solutions for real world problems. Constructive research happens in cycles with two phases. One phase is construction of the system and the other the evaluation of that system. The idea is to develop artifacts with potential practical value and also knowledge of the actual performance and value of these artifacts [18].
"Constructive research method implies building of an artifact (practical, theoretical or both) that solves a domain specific problem in order to create knowledge about how the problem can be solved (or understood, explained or modeled) in principle. Constructive research gives results which can have both practical and theoretical relevance" [14].
When applied to a problem, the methodology of constructive research pro-

duces innovative constructs. These innovative constructs can be theory, algorithms, models, software or frameworks. In this way constructive research makes contributions to the theory of the discipline in which it is applied. It is also important to note that "constructive research can also be viewed as a form of conducting case research parallel to ethnography, grounded theory, theory illustration, theory testing and action research"[27]. We can say that the constructive research approach is one of the most important research methods in computer science. A neutral and critical attitude is expected from scientist during the overall process of development and implementation of an innovative construct [27, 17].

In constructive research, fuzzy information is collected from different information sources such as literature reviews, processes, training materials, working experience etc. which provides a theoretical body of knowledge. An innovative construct, theoretical framework or solution is extracted from the theoretical body of knowledge according to its relevance to the problem. This new construct or solution produces new knowledge to further extend the *practical* and *epistemic* boundaries of the knowledge. This overall idea is depicted in the Figure 4.1 [43].
In our particular topic of Human Machine Interaction, the artefact/innovative constructs to be developed are gaze-contingent interfaces for mobile robot tele-operation and the targeted knowledge is knowledge of human performance with these interfaces. In order to get the whole system working, some hardware and software components are also needed to integrate system components so as to work to get the desired functionality out of the system. Information about the usage of these components is extracted from the documentation provided with these components. The outcome of this overall activity will be in the form of a *prototype*.

4.2 Quantitative Research

We use an experimental strategy of inquiry, i.e. a *quantitative* research methodology for the validation of the innovative construct produced from the previous phase. In this approach the researcher specifies a narrow hy-

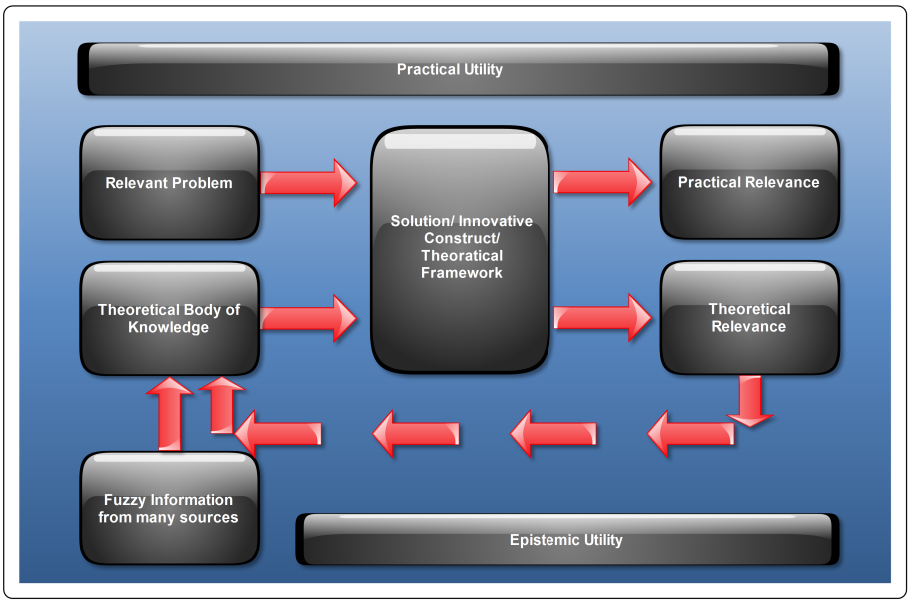

Figure 4.1: *Graphical representation of Constructive Research approach [43]*

pothesis and then tests the theory. Data collected as a result of successful experiment execution is used to support or refute the hypothesis. An experimental design is used and data is collected using instruments which measure outcomes as a result of different input factors. Information collected is then analyzed using statistical methods and hypothesis testing. We have used a *quantitative* research approach because in our study we can distinctly identify the best predictors (factors) that can influence the outcome [9].

4.3 Application of Methods in this Thesis

In this section we explain how the selected method should address all of the research questions described above. We discuss this question by question in order of the research questions.

4.3.1 Research Question-1

How can different components, an Eye Gaze Tracker, a Tele-operation Station and a Mobile Robot, be integrated with each other to work as a whole system?

"Only those people can lay new strong foundations who already know what actually went wrong with the older ones" [28]
The constructive research approach provides answer to this question. The step by step approach is [19]:

- Description of the problem and the solution. We already have discussed and defined the problem in chapter 3. Chapter 5 gives a full description of the solution.
    - ◇ New interface design is the core of the thesis. Literature review is the only way to find out what has been done so far in some specific area. So we have conducted an extensive literature review to find out essential details of what has already been done in our specific area. This gives us deep insight and knowledge about our topic and enables us to design a *new approach towards interface design*. This is discussed in length in the earlier part of chapter 5.
    - ◇ There are diverse hardware and software components involved in the development of the prototype. These are: Tobii Eye Tracker T60, GrabBee video grabber, Intel's image processing library OpenCV, the Arduino open-source electronics prototyping platform for mobile robot navigation and XBee serial communication transceivers. So it is very necessary to carefully analyze and select the tools and techniques needed to integrate these components. The later part of the chapter 5 describes all technical details of our implementation.
- Analysis of the solution.

- Comparison of the solution with alternate solutions.

4.3.2 Research Question-2

How can we improve the performance of gaze interaction by making different arrangements of the active regions of the operator interface?

Empirical evaluation is a kind of research which drives its data from direct observation or experiment. In our specific case we have designed two experiments to collect evidence regarding whether our developed solution improves the performance of the system compared to existing solutions by other researchers in this area. Chapters 6 and 7 tell the whole story.

4.3.3 Research Question-3

What are the limitations of gaze interaction, if any, and the reasons for these limitations?

Based on the analysis of evidence collected from empirical evaluation in research question 2, we discuss limitations of gaze interaction. Details are presented in chapter 8.

4.4 Graphical Overview of the Research Plan

Figure 4.2 presents a graphical overview of the overall research plan. Different stages of work are marked where we get answers to our research questions.

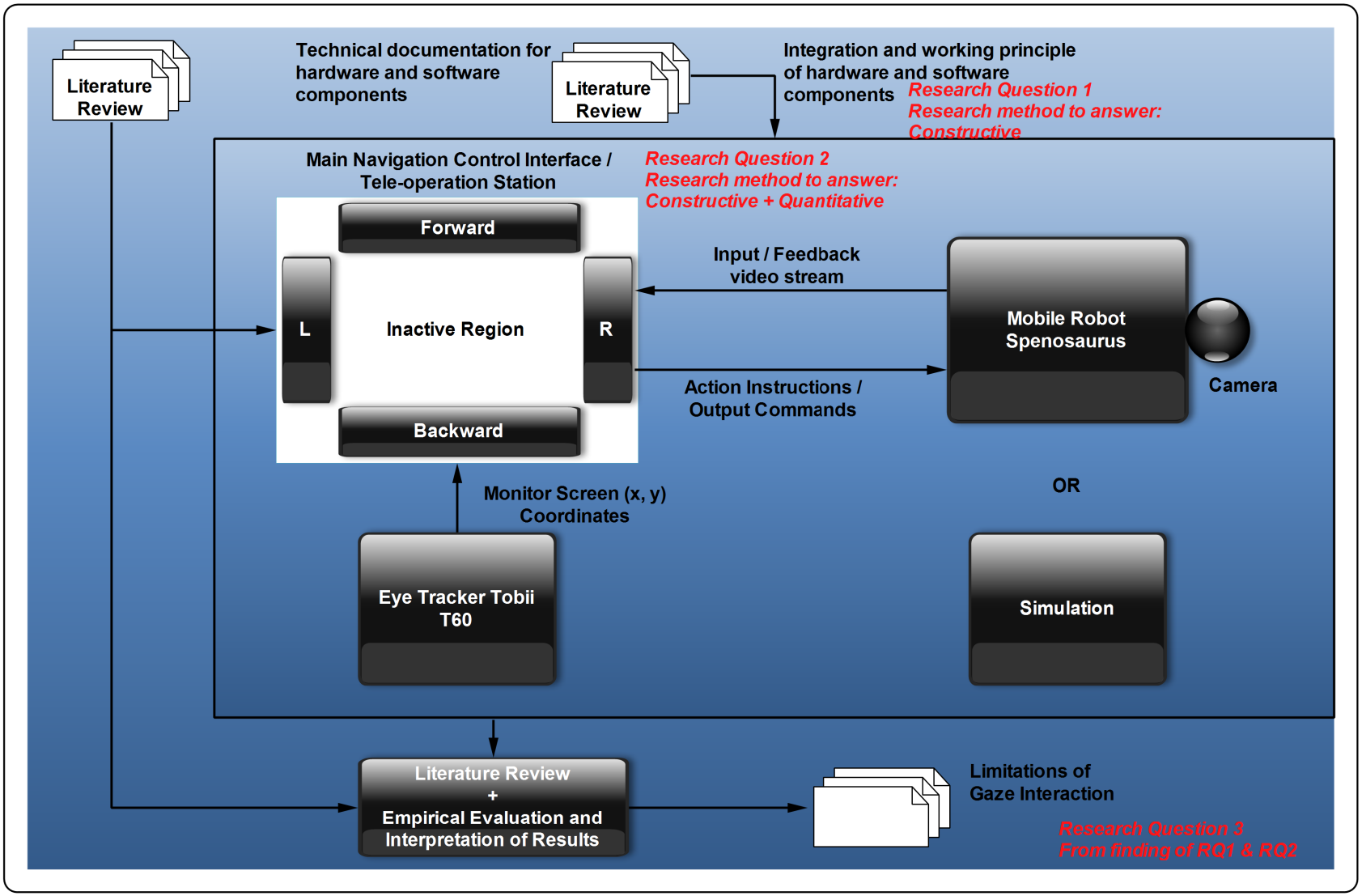

Figure 4.2: *Graphical overview of the Research Method*

Chapter 5

Theoretical Work

In this chapter we present an in depth study and analysis of different aspects of all the components to be used in the construction of our prototype. In the beginning of the chapter we present theoretical work which provides grounds for our new innovative dynamic interface. In the second part of the chapter we discuss integration aspects of different hardware and software modules. Both earlier parts of the chapter lay a foundation for software architecture for the prototype to be developed which will be used for evaluation purposes afterwards. In the last part of the chapter we present this software architecture.

5.1 Literature Review

Literature review provides the knowledge of state-of-the-art and really helps us to foresee and play around with alternative solutions in a well directed manner.

5.1.1 Evolution of the Eye Tracking Systems

Eye Gaze Tracking has deep roots in history. There was a proliferation of Eye Tracking Technologies by the end of 19th century and the beginning of 20th century. The purpose of the early developments in this field was to understand the nature of human eye movements. Delabarre in the late

1800s developed an eye tracking device using an eye cup. Extended from this eye cup was a lever to draw eye movements on a smoked drum. This technique was highly invasive and uncomfortable. Since the eye cup was directly attached to the eye surface. There was a hole in the middle of this cup through which the test subject could see [41]. Dodge and Cline later in 1901 developed a more comfortable way to record eye movements by using light reflected from the surface of the eye. They used photographic devices that require no direct attachment to the surface of the eye. Many modern Eye Tracking Systems are based on this principle [41].

During the last 3-4 decades, the main focus of eye tracking research has been on assistive technologies for people with different disabilities. These technologies provide a variable degree of assistance to the disabled according to the nature of their disability. Among these researches [15, 13, 29] mainly focus on *eye* or *gaze typing* and [32, 26] focus on *eye gaze* directed movement of wheelchairs.

5.1.2 Contemporary Technologies

Eye movement measurement methodologies can be divided into four broad categories: Electro-OculoGraphy (EOG), Scleral Contact Lens/Search Coil, Photo-OculoGraphy (POG) or Video-OculoGraphy (VOG) and video-based combined pupil and corneal reflection. The basic principle of the Electro-Oculo-
Graphy is dc-signal recordings of the electric potential differences of the skin surrounding the ocular cavity. The scleral Contact Lens/Search Coil method provides the most accurate eye movement measures. In this method a mechanical or optical reference object is attached to a contact lens and then this lens is worn by the test subject directly on their eye. POG or VOG involves the measurement of distinguishable features of the eyes under rotation/translation for example, the apparent shape of the pupil, the position of the limbus (the irissclera boundary) and corneal reflections of closely situated directed light source (often infra-red). A problem with all these techniques is that they usually do not provide point of regard measurement. A special

advantage of video-based trackers is their use of image processing hardware to calculate the point of regard in real time. The *Tobii T90* is a video-based eye tracker that we will use in our experiments [11].

5.1.3 Gaze Contingent Interfaces

As an example of gaze contingent interfaces for the disabled, [26] describes an interface developed to control a powered wheelchair. This interface is operated by human eye gaze. The interface is divided into 9 regions. Of those 9 regions, 4 are gaze contingent or command regions. Gaze contingent regions in an eye gaze controlled interface are those regions that are used to trigger specific commands when the eye gaze falls within them. In [26], screen regions are used to initiate wheelchair motion commands. Similar work is presented in [4], where electrooculography (detection of electrical impulses from muscles that control eye gaze direction) is used to send driving commands to a wheelchair. In both of these works, the operator is sitting on the wheelchair. No feedback is provided to the operator through the interface, i.e. the interface is used only for one-way communication.

An experimental eye-tracking algorithm has also been used to control a robotic arm [47]. For this experiment, the interface is divided into 2 regions: a command region and the feedback region. Feedback is provided in the form of images taken by a camera installed in the robot location. Similar work has been presented for control of a robotic toy [6]. In this work a different eye-tracking technique is used: in order to find out the gaze direction, the location of the eye ball in the eye socket is tracked. In this technique the only purpose of the interface is to present feedback to the operator.

Minimal Invasive Surgery (MIS) is growing in its popularity. Due to its popularity there is a demand to improve its functionality and usability. DaVinci (a surgical robot) is an existing robotic assisted MIS technology (Intuitive Surgical, Sunnyvale, CA) that allows the surgeon to interact with the operative environment remotely through a tele-operation station. In this system the hand movements of the surgeon are replicated by specialized instruments. In such a configuration it is not possible for the surgeon to use

more than two tools at one time even when additional tools are available. The surgeon needs assistance of another surgical member in this scenario or rather he can relinquish the control of one instrument and switch to the other. In this particular scenario David P. Noonan et al. (Imperial College London) have presented a *gaze* contingent framework whereby an additional tool can be controlled directly by the eyes of the surgeon [30].

In the more general field of robotics (i.e. beyond concerns with human disability) most research addresses controllable agents rather than fully autonomous agents [31]. In most cases these controllable agents are required to be controlled from a remote location, an approach called tele-operation [23]. TeleGaze [23] and Gaze-controlled Driving [36] are recent projects in the area of robotics and control systems using eye gaze contingent interfaces for more natural human robot interaction. In TeleGaze, the interface is divided into 9 gaze contingent regions for different commands. These gaze contingent regions take 1/3 second dwell time to activate and issue the associated command to a WIFI enabled modified wheelchair robot. Dwell time enables eye gaze to act as mouse click. When the operator fixates in a gaze contingent region for a predetermined interval of time (e.g. 1/3 sec), it is considered as an activation similar to a mouse click and a command is issued to the system to be controlled.

Tele-operation is characterized by controlling some system from a remote location [23]. It involves two separate actions: one is to monitor the current state of the system to be controlled and the second is to send the appropriate commands according to the current state of the system. Monitoring is typically performed by the human eye and the hands are responsible for command execution through conventional input modalities like a key board, mouse or joy stick. If eye gaze alone is used as an input modality for tele-operation, both monitoring and command execution must be performed by eye gaze. The eye can fixate (i.e. focus at a point on the screen to keep a cursor stationary at a gaze contingent region for a certain interval of time) [22] only on one object at a time; i.e. when the operator tries to focus on feedback (Monitoring) she loses attention on the command execution interface, and vice versa (this applies in all of [26], [4], [23] and [22]. The resulting

constant switching between two areas of attention decreases the performance of the system considerably. The main reason for this performance deterioration is dwell time. Dwell time reinitializes to zero if the operator loses focus within a gaze contingent region in the interface. So switching between feedback-monitoring and command-execution takes repeated dwell time to send commands, resulting in performance deterioration.

An alternative design, described in next section, is that if an operator fixates in a gaze sensitive region and after the dwell time, the whole feedback region becomes the command execution region as well, then the overheads of switching between command execution and feedback regions is eliminated. This is explained in detail in next section.

5.2 A New Approach to Interface Design

We seek an answer to the following research question in this thesis: How to avoid multiple unnecessary dwell times and improve performance of the overall activity of tele-operation in a gaze-directed interface? In order to answer this question we decided to analyze gaze behavior of participants on a similar kind of interface proposed in related work studies [23], in tasks controlling a wheeled mobile robot with visual feedback provided by a real time video link from a camera mounted on the robot platform. Gaze-directed control areas of the screen are mapped over the user's view of the video transmission from the robot. The robot can be driven in forward and backward directions, or turned by control of its differential drive system. Figure-1 is a scatter plot of a user's gaze fixations in different parts of the screen while interacting with an interface using fixed areas for dwell-activated command initiation. The participants have used all the gaze contingent regions according to their needs. An interesting observation is that participants have tended to prefer the upper half areas of the RIGHT and LEFT regions of the interface. These are black and blue regions in the plot, respectively. We used this observation in our alternative interface design and shortened these regions by 70 pixels from below, using this region instead for the STOP command in our alternative interface (Figure-2). Gray portions in the plot represent all those

fixations when the participant focuses in the feedback region or some other regions in the monitor screen than gaze contingent command regions. In the new design, the STOP region dynamically adjusts its position in the interface. Initially it rests on top of the BACKWARD region. If a participant fixates in the BACKWARD region it changes its position and sticks to the FORWARD region at the top of interface. Whenever user switches between FORWARD and BACKWARD command regions, the STOP region moves in between, since it is logical to stop before moving forward or backward. Our interface consists of 10 gaze contingent regions in total: 2 regions for STOP, 4 regions for FORWARD, BACKWARD, RIGHT and LEFT. The remaining 4 are dynamically formed by the expansion of the initial FORWARD, BACKWARD, RIGHT and LEFT regions. All other regions except STOP expand over the whole feedback region after the dwell time when fixated by the user. This provides more ease to the user since it is easy to fixate in comparatively larger regions. It also eliminates repeated dwell times for multiple commands. When one task is completed, the user fixates in some other gaze contingent region to perform a task associated to that region. As a result, the previous gaze contingent region shrinks back to its default location and the new region as selected by the user merges with the feedback region, and so on. This is a more appropriate technique for tasks that need multiple commands of the same type (e.g. turning to a suitable angle towards the right in multiple increments of 15°).

5.3 Hardware Specifications

5.3.1 Tobii T60© Eye Tracker

The Tobii T60 is a video-based eye tracker that uses corneal reflection to calculate gaze points on the eye tracker display screen. It uses infrared diodes to generate corneal reflection patterns on the cornea of a user's eye. Image sensors collect other necessary visual information about the person along with corneal reflection patterns. This visual information is processed using image processing algorithms to identify relevant features, including the eyes and the

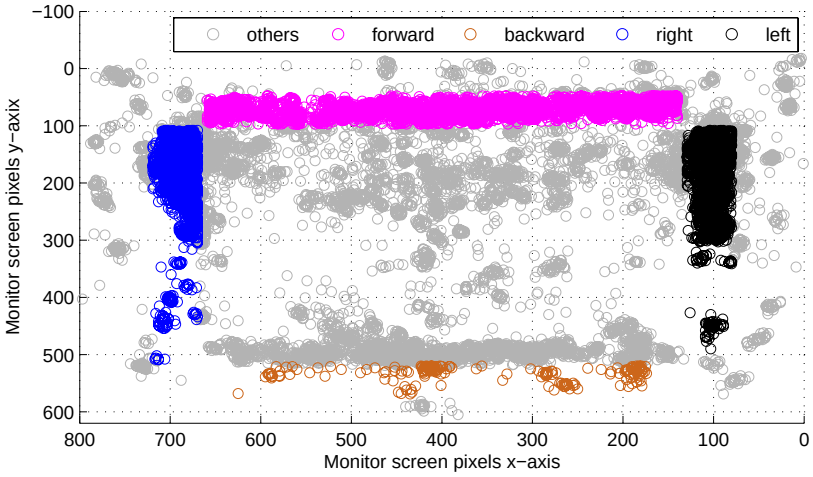

Figure 5.1: *Gaze behaviour of the user.*

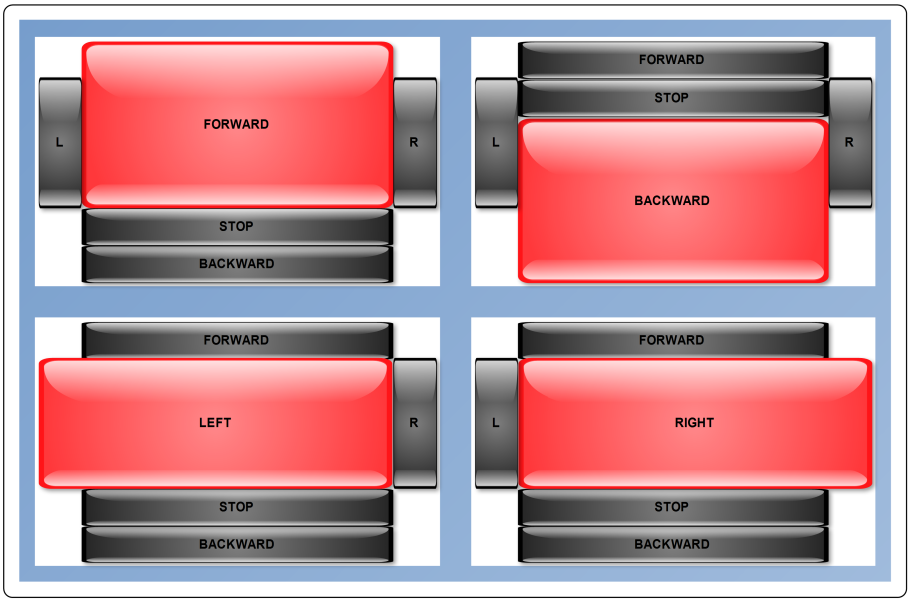

Figure 5.2: *Four different states of the interface.*

corneal reflection patterns. Based upon this information, the position of the eye ball is calculated, and then the gaze point on the screen, that is, where the user is looking. Technical specifications for Tobii T60 are given in Table 5.2 [37].

Accuracy: There is a difference between the Measured Gaze Direction and

Table 5.1: *Sample Gaze Behavior Data*

$x-coordinate$	$y-coordinate$	$Direction$
691	150	R
676	147	R
681	149	R
126	171	L
124	174	L
349	088	F
355	090	F
537	523	B
572	522	B
192	447	A
188	470	A
...	...	...

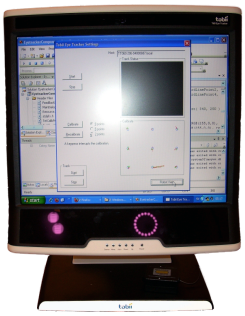

Figure 5.3: *Tobii T60© Eye Tracker [37]*

the Actual Gaze Direction at different parts of the screen for a person positioned at the center of the eye tracking box (i.e. the spatial volume from within which eye-tracking is effective). Drift effects and compensation errors are not included in this measurement. Varying external conditions such as lighting, quality of the calibrations and individual eye characteristics effect the accuracy of the Tobii eye trackers.

Drift: Drift is a measure of change in accuracy due to the change in lighting conditions. The specified drift value relates to complete inversion of screen color, e.g from black to white without recalibration in between.

Freedom of Head Movement: Freedom of head movement is a measure of the box (height x width in cm) where at least one of the eyes is within the field of view of the eye tracker. According to Tobii documentation, the stated value was measured at 70 cm distance from the sensor.

Table 5.2: *Tobii T60© Eye Tracker Technical Specifications [37]*

Description	*Specification*
Accuracy	0.5°
Drift	< 0.3°
Freedom of head movement	44x22x30cm
Data Rate	60Hz
Binocular tracking	Yes
Bright/dark pupil tracking	Both - automatic optimization
TFT Display<	17" TFT, 1280x1024Pixels
Eye tracking server	Embedded
User camera	Built in

Data Rate: The number of sampled gaze points per second. In our particular case, the Tobii T60 data rate is 60Hz that is 60 gaze data points per second are collected for each eye (one gaze data point after each 15ms).

5.3.1.1 Tobii Software Development Kit (SDK)

Tobii provides a Sofware Development Kit (SDK) with its eye trackers. Using this SDK we can develop application software which to provide customized routines to control and retrieve data from Tobii eye trackers. Due to the diversity in applications, different applications require different levels of interfacing. The Tobii SDK provides interfaces on different levels. It provides low level interfaces for those applications which need more customization while for applications which need less customization high level interfaces are provided. Along with other interfaces, the SDK also provides Microsoft Windows COM based interfaces. In our particular case we use COM interfaces to get required gaze data from eye tracker [37].

5.3.2 Spinosaurus Mobile Robot

The Spinosaurus (Figure 5.4) is a small mobile robot that has been developed as a platform for research in mobile robot control and human/robot interaction. The first version of the Spinosaurus (1.0) was completed in April 2009. This version was a pure radio controlled system, using FM analog control from a conventional hobby RC controller to steer and drive the robot.

A video camera mounted on the steerable turret of the robot sent an image to a receiver that was connected to a computer equipped with a Tobii T60 eye-tracking system. This configuration was used in a simple eye-tracking study to investigate human vision while remotely controlling the robot in a task to search for objects in a maze, using only the view from the robot camera as a guide.
In 2010 the Spinosaurus has been upgraded to version 2.0 that includes:

- removal of the analog FM radio control system
- addition of a bi-directional data link based upon Xbee wireless data transceivers
- the addition of 2 x on-board Arduino boards for sensor data acquisition, telecommunications handling and motion control
- The addition of several IR and ultrasonic distance sensors and an inertial navigation system for autonomous operations (not used in this study)

The data link can be used to control the robot from an external computer. See appendix A for details.

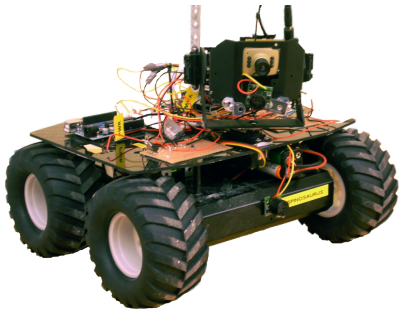

Figure 5.4: *Spinosaurus Mobile Robot used in experiment*

5.3.3 Arduino© Prototyping Platform

The Arduino© Duemilanove (Diecimila) is a micro-controller board. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog

inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button (Table 5.3). This is a programmable board that can be programmed with the Arduino software. The Arduino open-source environment can be used to write and upload programs to Arduino I/O boards [1].

The Arduino program used in this project responds to the commands received through a serial interface. It waits for the input command from the serial port and acts according to the pre-programmed behavior for each command. There are four servo channels connected via Arduino pins 8 to 11. The servo outputs driving the mobile robot turret servos go directly to those servos. However, the speed control outputs go to two control channels of a TREX speed controller (described in next section). Table 5.4 shows the movement configuration for the motors. There is certain way to communicate with the Arduino code through the serial interface. For every command character (U/D/R/L/A/S/W/X), a second character sequence must be followed that determines the speed/angle of the operation being requested. The speed has a range of 0-9 where 0 stands for no change and 9 accelerates the servo at the highest available speed. The angle has a range 0°-360°. In our particular case we use a small angle of 15° because it is hard to control the robot movement using larger rotations.
For example:
W9 (Accelerate with full speed).
S15 (Turn towards the right with a set turn angle i.e. 15°).

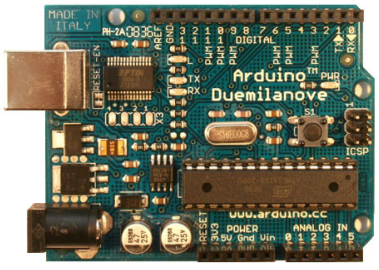

Figure 5.5: *ArduinoDuemilanove: A prototyping platform [1]*

Similar navigation protocol will be used for conventional input modalities

Table 5.3: *Arduino© Specifications Summary [1]*

Description	*Configuration*
Microcontroller	ATmega168
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	16 KB (ATmega168) or 32 KB (ATmega328) of which 2 KB used by boot loader
SRAM	1 KB (ATmega168) or 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) or 1 KB (ATmega328)
Clock Speed	16 MHz

Table 5.4: *Arduino© Navigation Configuration*

Function	*Movement*	*Keys(Charactertosend)*
Camera steer left/ Camera steer right	0°-90°/90°-180°	L/R
Camera tilt up/ Camera tilt down	0°-90°/90°-180°	U/D
Accelerate/ Decelerate	0-9	W/X
Move right/ Move left	0°-15°	S/A

i.e. keyboard, mouse and joystick.

5.3.3.1 Polou Trex Dual Motor Controller

Movement actuators of the mobile robot are motors having their speed and direction controlled by voltage outputs from a "Polou TReX Dual Motor Controller". Figure 5.6 shows a description of different components of "Polou TReX Dual Motor Controller". The TREX controller board can receive serial or PWM signals in 5 channels in order to drive two motors. Independent and combined movement of the channels is possible. The Arduino micro controller sends commands with appropriate polarity to the speed controller board and it controls the motors using voltages set according to the commands. Varying voltage and appropriate polarity control the speed and direction of the motors respectively [34].

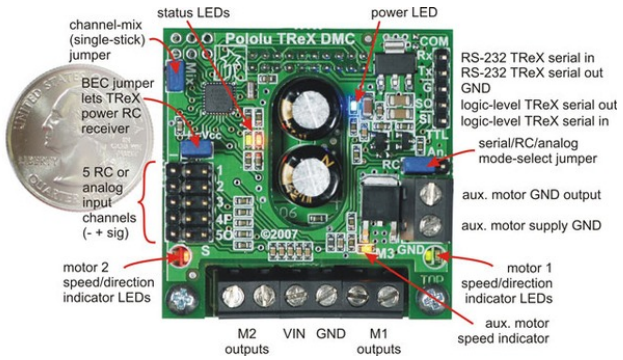

Figure 5.6: *Pololu TReX Dual Motor Controller [39]*

5.3.3.2 Arduino XBee Shield

The Xbee shield allows an Arduino board to communicate wirelessly using the Zigbee standard. It is based on the Xbee module from MaxStream. The module can communicate up to 100 feet indoors or 300 feet outdoors without obstacles [2].

Figure 5.7: *Arduino XBee Shield Module for Wireless Communication [2]*

5.3.4 XBee Serial Communication Module

Xbee serial communication module is a wireless data transceiver. In our work we establish a bidirectional data link between a remote computer and mobile robot using these transceivers. On the mobile robot side an Arduino micro controller is programmed to transmit and receive data across serial interface, while on the remote computer side this functionality is achieved using a C++ module written specially for serial communication.

Figure 5.8: *XBee© Module for Wireless Communication [45]*

5.3.4.1 XBee© Explorer Module

This is a simple to use, USB to serial base unit for the XBee line [46]. On the remote computer side, the explorer module is used to connect the Xbee transceiver to the computer using a USB cable.

Figure 5.9: *XBee Explorer Module for PC side Interface [46]*

5.4 Software Components

5.4.1 OpenCV©

OpenCV is an open source (see http://opensource.org) computer vision library available from http://SourceForge.net/projects/opencvlibrary. The li-

brary is written in C and C++ and runs under Linux, Windows and Mac OS X. There is also active development on interfaces for Python, Ruby, Matlab, and other languages. OpenCV was designed for computational efficiency and with a strong focus on realtime applications. OpenCV is written in optimized C and can take advantage of multicore processors. For further automatic optimization on Intel architectures, Intel's Integrated Performance Primitives (IPP) libraries [IPP], which consist of low-level optimized routines for many different algorithmic areas, can be used. OpenCV automatically uses the appropriate IPP library at runtime if that library is installed [8].

5.4.2 Coding Language Selection

After detailed study of hardware and software components to be used in the prototype we can summarize as follows:

- The Tobii Eye Tracker SDK provides ATL-COM (Active Template Library-Component Object Model) objects to provide real time gaze data in the form of x, y coordinates of user gaze points on the screen at a data rate of 60Hz.

- The XBee Wireless Communication Module provides a serial link to send the commands to the Arduino board on the Spinosaurus robot. C++ is the most efficient language to drive the serial link from the application system.

- OpenCV is an open source image processing tool from Intel. It is a C++ based tool. It can be used to grab the video stream sent wirelessly from the camera mounted on the Spinosaurus and then to augment this stream with our new innovative interface elements.

Whole scenario suggests the use of the C++ language for prototype application development. By using C++ we avoid the need for middle-ware which may result in less efficient functionality due to different software layers among diverse end points.

5.4.3 Interface Implementation Module

After analyzing all hardware and software components to be used in the prototype, we came to the conclusion that using C++ we can integrate all components to get our aims and objectives more easily than the use of other languages. We used OpenCV in our prototype for two purposes: to grab the video stream from the camera mounted on the mobile robot, and to create a gaze contingent interface by augmenting the robot view video stream.

5.5 Prototype Architecture

A software architecture is a structural plan which is used as a blue print during the development process. It describes different elements of a system that how they can be used together to fulfil the requirements of the system. Software architecture is an abstraction that helps manage the complexity of system development. It is not a comprehensive decomposition of the details. Many of the details may be hidden in an abstraction of the architecture [16].

5.5.1 Conceptual View

A conceptual view is closely related to the application domain. Different architecture elements are used in this view. Conceptual components show different functionalities, while data exchange and coordination is shown by connectors. Figure 5.10 shows a conceptual view of our prototype [16]. Here the conceptual view describes:

- How the system fulfils its requirements.
- How the commercial off-the-shelf components are integrated and how they interact with rest of the system.
- How domain-specific software and hardware is incorporated into the system.

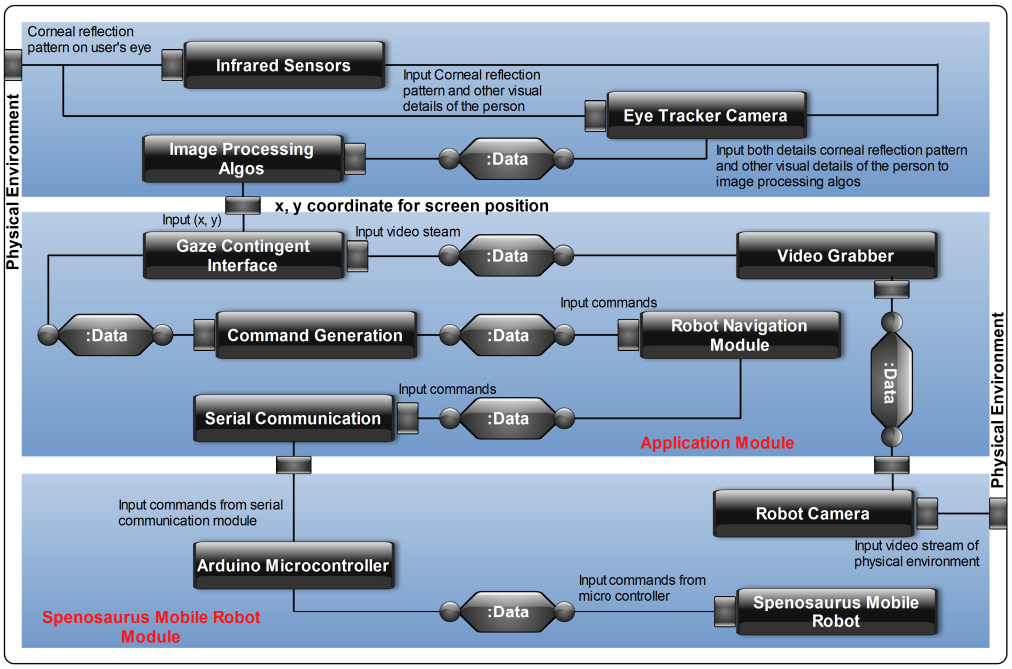

Figure 5.10: *Conceptual view of prototype*

5.5.2 Module View

The module view describes how the conceptual view has been realized by using specific software tools and technologies. Components and connectors from the conceptual view are mapped to subsystems and modules. Figure 5.11 presents this mapping [16].

5.5.3 Execution View

Purpose of the execution view is flow of control. The conceptual view describes the logical flow of control while in the execution view one is interested in the flow of control in terms of the implemented runtime platform. Figure 5.12 presents this flow of control [16].

5.6 Application source code

See Appendix B for Application source code.

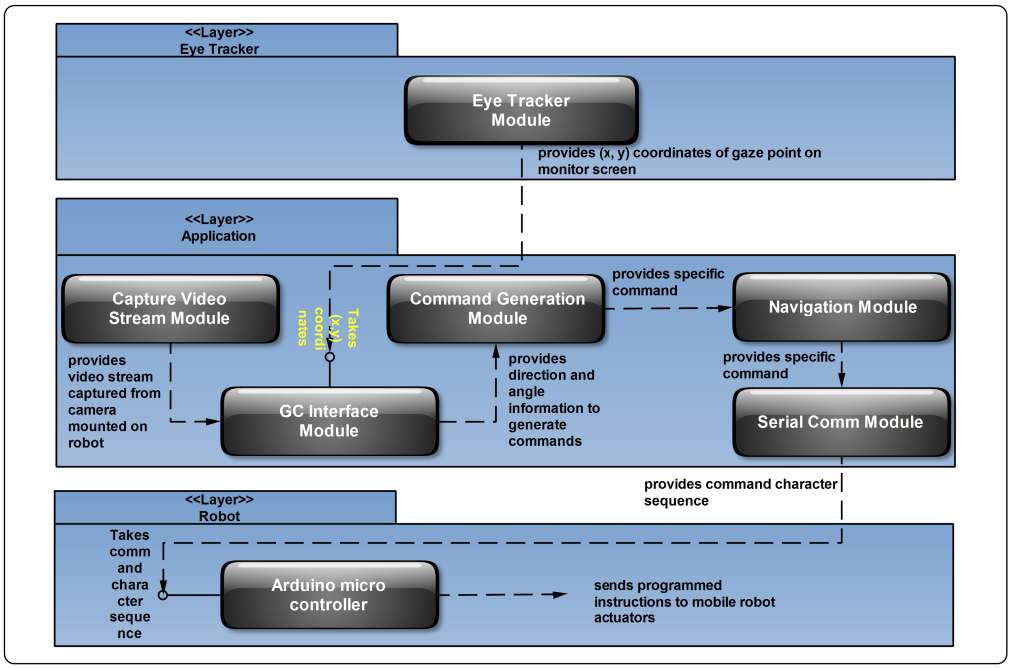

Figure 5.11: *Module view of prototype*

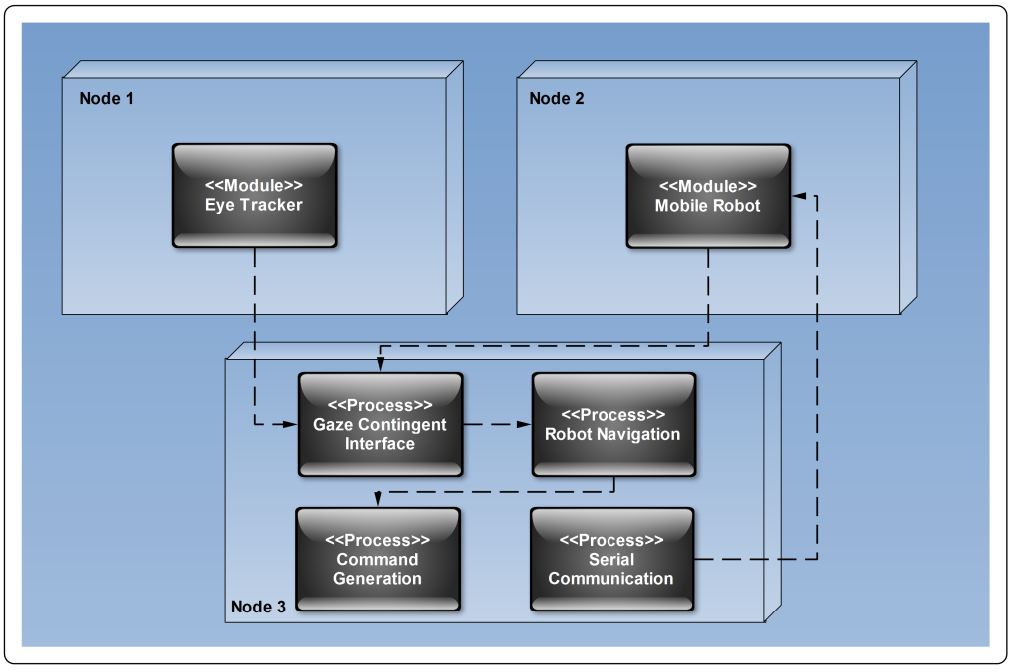

Figure 5.12: *Execution view of prototype*

Chapter 6

Empirical Evaluation

In this chapter we present the empirical evaluation of the prototype and analysis of the results. In the particular case of our thesis, the objective of our empirical study is to determine the difference in individual performance of different input modalities while they are applied to control mobile robot navigation (for the Spinosaurus robot). These modalities include: keyboard, mouse, joystick, a static gaze-contingent interface[23] and a dynamic gaze-contingent interface. Our main interest is the evaluation of our newly developed and novel dynamic interface. We need to understand that where it stands in relation to the other modalities.

Object of the Study: The object of the study is to evaluate the input modalities for remotely controlling mobile robot navigation in terms of their individual performance for completing a particular task.

Purpose: The purpose of the experiment is to evaluate the individual performance of each input modality for controlling mobile robot navigation. The outcome of the experiment outcome provides insight into how gaze-contingent interfaces are different from conventional input modalities, and their relative performance. It also enables us to understand some limitations of using gaze as an input modality in comparison with conventional input methods.

6.1 Experiment Definition and context

Our empirical study is based on two experiments. The experiments are online and real time using the prototype developed in the previous phase. The ability to generalize from this particular context is further elaborated in the analysis phase. Both the experiments are task-oriented. Human robot interaction applications are very diverse. Hence, there are no standard metrics for evaluation of newly developed applications [23]. However, there are some common metrics in any application domain that are most likely to be used to evaluate the application developed in that particular domain [38]. Time needed to complete a task is a common measure [40], and this measure is adopted here.

Experiment 1: Experiment 1 is a rather small experiment. The task in this experiment is to turn the robot at an angle of 90°. The reason for this task as a separate experiment is that the robot we are using can turn only with a fixed angle of 15° per turn command. In order to get larger turn angles, multiple turn commands are needed; e.g. with a command turn angle of 15° the robot needs 6 turn commands to turn a total angle of 90°. Forward and backward movements are far simpler than turning, since forward and backward motion is continuous. That is why we decided to examine the turning motion in a separate experiment.
The upper half of Figure 6.1 shows the experiment 1 setup. A green coloured card is placed in front of the robot. Another card of red color is placed on the left side of the robot at an angle of 90° from the green card. When a participant fixates in the gaze contingent region specified for LEFT, the robot starts turning to the left. When the turning robot completes an angle of 90° it comes in front of the red card. Now the participant can see the red card in the feedback region of the interface and should gaze in the stop region of the interface to stop further movement. The participant performs this activity with all input modalities including static and dynamic versions of the interfaces.

Experiment 2: In experiment 2, a track is designed for the robot to travel on. An experiment participant interacts with the robot using all input modalities that is, both gaze-contingent interfaces (static and dynamic) and conventional modalities (key-board, mouse and joystick). The experimental task is to navigate through this track. Participants perform both of the experiments multiple times. Data is collected by noting the task completion time for each experiment trial. In experiment 2, participants are given a task to send commands to the robot and monitor visual feedback, to drive the robot around a specified track to return to the starting location.
The lower half of Figure 6.1 presents the layout of the track used in experiment 2. Experiment 2 is designed to include all possible navigational moves.

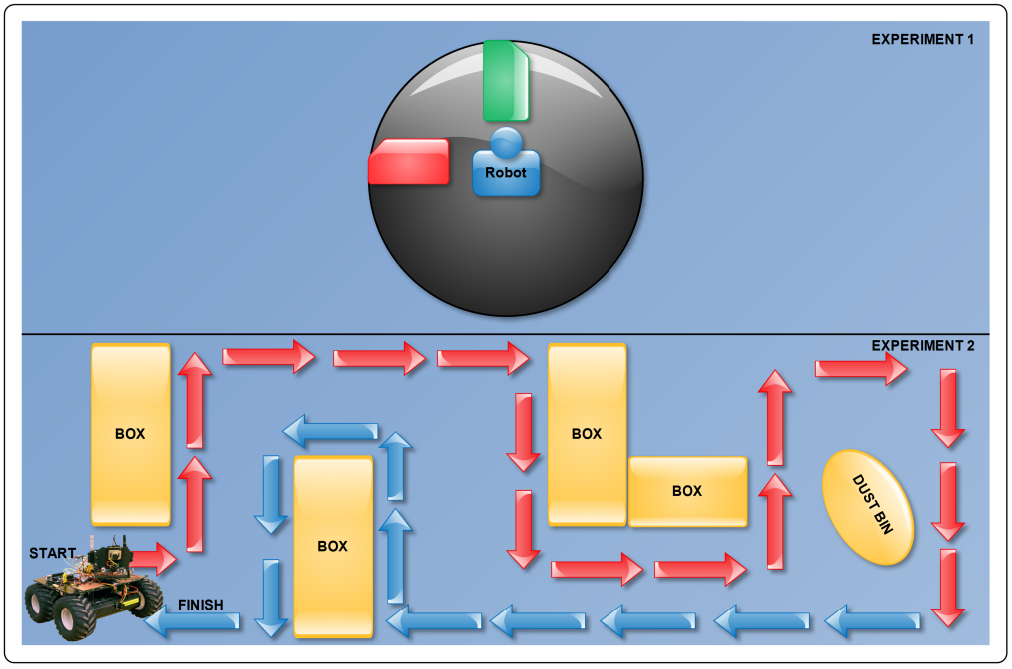

Figure 6.1: *Task layout for experiment 1 and 2*

The total length of the track is 19.25 meters. The start and finish points are same. The whole track is marked with red and blue arrows. A participant follows the red arrows to move down the track to the turning area and the blue arrows for coming back to the finish point.

Table 5.4 shows navigation configuration of the arduino micro-controller. It takes W, X, A and S characters for forward, backward, left and right navigational movements respectively. For keyboard, same keys (W, X, A and S) are mapped for the respective navigational movements. Similar protocol is used for the joystick as well. For mouse, already developed static interface is used for generating commands. When operator clicks in FORWARD region, command for forward movement is generated and sent to the micro-controller unit. Other commands (BACKWARD, LEFT, RIGHT and STOP) are generated in similar way. STOP command can be generated by writing W0 or X0 on serial port. Digit zero with W and X means zero acceleration.
15 participants in total took part in these experiments. In experiment 1, 10 trials were recorded for each participant. This results in 15 x 10 = 150 trials of data. Experiment 2 is a bit lengthier in terms of time. Hence a block of 3 trials is recorded for each participant, giving 15 x 3 = 45 trials in total.

6.1.1 Hypothesis Formulation

Before the design and execution of an experiment it is very important to know and state clearly what we intend to evaluate in the experiment [44]. This leads us to the formulation of a hypothesis.
In our particular case, as we know from the problem definition presented in previous chapters that performance of static interfaces for gaze-directed control is limited due to repeated dwell times, so our newly proposed dynamic interface is proposed to significantly improve performance by eliminating repeated dwell time.
This informal statement of hypothesis can be stated more formally as follows, including evaluation measures.

Null hypothesis, H_0: There is no difference in performance (measured in terms of task completion time) between the Static Gaze-Contingent Interface (SGCI) the and Dynamic Gaze-Contingent Interface (DGCI).
H_0: Performance(SGCI) = Performance(DGCI).
Alternative hypothesis, H_1: Performance(SGCI) < Performance(DGCI).

Measures needed: Performance (Task completion time).

6.1.2 Variable Selection

In our empirical study the independent variable is screen adaptiveness, which has two possible values of *static* and *dynamic*. These interface variants are used to control the navigation of a mobile robot. In both experiments participants complete a navigation and movement task. The outcome/dependent measure of the experiments is task completion time.

6.1.3 Participants/Users

Fifteen participants took part in this experiment. All were from Blekinge Institute of Technology (BTH). Out of fifteen, 10 were male and 5 female. Two faculty members also took part in this experiment. The age range of the participants was 20-51 years. In order to ensure that all the participants had the same level of understanding they were briefed in the same way about the experiment. They were also each given a training session before data collection started.

6.2 Experiment Design

The purpose of an experiment is to draw meaningful conclusions regarding the problem at hand. Statistical analysis methods are applied to collected data in order to interpret the results. To get real advantage of an experiment it is very necessary that the experiment is carefully planned and designed. The application of a particular statistical technique depends on the experiment design chosen and the measurement scales used [44].

6.2.1 General Design Principles

General design principles of Randomization and Balancing are used in our experiment [44].

6.2.1.1 Randomization

Subjects or participants for experiment execution are selected randomly. Both genders are present among the participants. Representatives of different disciplines in the form of students and teachers of different departments were selected.

6.2.1.2 Balancing

Balancing simplifies and strengthens the statistical analysis of data. We use the same number of subjects or participants for every treatment in the experiment, that is, same number of persons in each group.

6.2.2 Design Type

In our experiment we investigated the performance of navigation for controlling a mobile robot using diverse input modalities. The factor involved in our experiment is input modality, where different treatments available are: mouse, key-board, joystick, the static gaze-contingent interface and the dynamic gaze-contingent interface. For these kinds of experiments the appropriate design type is *one factor with more than two treatments*, and a suitable analysis technique is ANOVA [44].

6.2.3 Apparatus and Conditions

An Intel© Core 2™ Extreme Q6850 @3.00GHz was programmed using Microsoft© Visual C++™ to create the gaze-directed monitoring and control interface. A Tobii© Eye Tracker T60™ was used to get participant's gaze behaviour and this was integrated with the interaction system via an API in order to use gaze data on the monitor screen. A GrabBee© video grabber was used to capture the video stream from the wireless camera mounted on the head of the mobile robot. Intel's© image processing library OpenCV© was used to grab the video stream and then superimpose interface components on it. The Arduino© open-source electronics prototyping platform was used to configure and map robot actuation via different commands. XBee© serial

communication transceivers were used to communicate commands between the user application software and the mobile robot. The interfaces developed in the project were able to work on any resolution but all experiments were conducted with 800x600 screen resolution.

6.2.4 Procedure/Execution

When participants arrived they were briefed about the experiments so that they had an idea about the whole activity to be performed. Participants were seated at a 60-65cm viewing distance from the eye-tracking device as directed in the Tobii© T60 user's manual. A calibration process was performed for each participant to get accurate gaze fixation points for each participant on the eye-tracking screen. Then eye tracking was started. Each participant was asked whether the cursor moved to the desired location on the screen according to her gaze fixations or not. When the participant was satisfied with the result, the robot camera view with the interface superimposed upon it was presented to the participant. This concluded the experiment setup, after which the participant could fixate in the gaze contingent regions to send commands to the robot. Each participant was allowed 3-4 training sessions to gain familiarity with the environment. Outcomes for actual trials were then recorded according to the pre-planned requirements.

Outcomes for three conventional modalities i.e. keyboard, mouse and joystick were also recorded. No training was provided for these modalities because participants were already familiar and comfortable with these input modalities.

While conducting experiments, input modalities were not used in a specific order. Specially in case of the Static and the Dynamic interfaces, they were used in different order for each participant. It was done intentionally to avoid biased results.

6.2.5 Validity Evaluation

We observed that the majority of the participants were not familiar with the concept and technology of eye tracking. This was a potential valid-

ity threat and could have affected the outcome due to varying capabilities and understanding of the participants. To bring them to an equal level of understanding we arranged training sessions for the participants. Another validity threat was power supplies for the logic and actuation parts of the mobile robot. Weak batteries may result in slow movement and delayed responses. To avoid this threat all the batteries were replaced with a fresh and fully charged set of batteries before switching to the next participant. XBee transceivers were used for communication between the application program and the mobile robot, having a range of 120 meters without obstacles. We ensured that the mobile robot remained within this range while performing tasks.

6.2.6 Data Sets

Tables 6.1 and 6.2 present the data collected as a result of experiment 1 and 2 execution.

Table 6.1: *Experiment 1 Data Set*
Task Completion Time in Seconds

Replicate	*Dynamic*	*Static*	*Mouse*	*Keyboard*	*Joystick*	*RowTotal*	
1	2.9	3.5	2.46	2.51	2.47	13.84	
2	2.8	5.7	2.45	2.49	2.49	15.93	
3	3.2	4.4	2.49	2.49	2.49	15.07	
4	3.1	3.4	2.46	2.46	2.49	13.91	
5	3.1	3.8	2.49	2.49	2.49	14.37	
6	3.5	3.2	2.49	2.46	2.52	14.17	
7	3.0	3.6	2.49	2.49	2.49	14.07	
8	2.8	3.8	2.49	2.49	2.49	14.07	
9	2.8	4.5	2.49	2.49	2.49	14.77	
10	2.8	4.4	2.49	2.49	2.49	14.67	
11	3.1	4.3	2.49	2.49	2.49	14.87	
12	2.7	4.5	2.49	2.49	2.49	14.67	
13	2.6	5.3	2.49	2.49	2.49	15.37	
14	2.4	4.6	2.49	2.49	2.46	14.44	
15	2.5	5.3	2.49	2.49	2.51	15.29	
16	3.0	4.9	2.49	2.49	2.49	15.37	
17	2.4	4.1	2.46	2.49	2.49	13.94	
18	3.2	4.8	2.46	2.49	2.49	15.44	
19	2.9	4.9	2.46	2.46	2.5	15.22	
20	2.6	3.4	2.46	2.49	2.49	13.44	
21	2.7	4.1	2.49	2.49	2.49	14.27	
22	3.1	2.8	2.46	2.49	2.49	13.34	
23	2.7	2.7	2.49	2.45	2.49	12.83	
24	3.1	4.8	2.49	2.49	2.49	15.37	
25	3.2	3.6	2.49	2.49	2.49	14.27	
26	3.1	2.9	2.48	2.46	2.48	13.42	
27	3.2	3.0	2.49	2.49	2.49	13.67	
28	3.0	3.8	2.49	2.49	2.49	14.27	
29	2.7	3.0	2.49	2.49	2.49	13.17	
30	2.8	5.3	2.46	2.48	2.46	15.5	
31	2.8	4.1	2.49	2.49	2.49	14.37	
32	2.8	4.4	2.49	2.49	2.49	14.67	
33	3.0	3.8	2.47	2.49	2.49	14.25	
34	2.9	3.2	2.46	2.46	2.49	13.51	
35	2.9	3.9	2.49	2.49	2.49	14.27	
36	3.2	3.8	2.52	2.49	2.49	14.5	
37	3.6	3.8	2.49	2.49	2.49	14.87	
38	2.9	3.1	2.49	2.49	2.49	13.47	
39	3.1	3.7	2.51	2.51	2.47	14.29	
40	3.2	4.1	2.49	2.49	2.49	14.77	
41	2.7	3.1	2.49	2.47	2.46	13.22	
42	3.0	5.4	2.49	2.49	2.47	15.85	
43	2.8	4.2	2.49	2.51	2.49	14.49	
44	3.3	4.1	2.46	2.49	2.46	14.81	
45	3.2	3.8	2.49	2.49	2.49	14.47	
46	3.6	5.4	2.46	2.49	2.48	16.43	
47	3.2	3.9	2.49	2.47	2.49	14.55	
48	3.2	5.2	2.49	2.49	2.49	15.87	
49	3.1	2.9	2.49	2.49	2.47	13.45	
...	...	...	...	...	...	...	
145	2.6	3.4	2.49	2.46	2.49	13.44	
146	2.6	3.7	2.49	2.49	2.49	13.77	
147	2.9	2.9	2.49	2.45	2.51	13.25	
148	2.4	3.0	2.49	2.45	2.49	12.83	
149	2.8	2.8	2.52	2.45	2.47	13.04	
150	2.7	3.3	2.49	2.49	2.49	13.47	
Sx	445	596.5	373.03	372.52	372.79	2159.84	Grand Total
n	150	150	150	150	150		
$\overline{x}$	2.966666667	3.976666667	2.486866667	2.483466667	2.485266667		
Sx^2	1329.28	2456.67	927.7069	925.1786	926.5033		
$(\mathrm{Sx})^2$	198025	355812.25	139151.3809	138771.1504	138972.3841		
Sd^2	9.113333333	84.58833333	0.031027333	0.037597333	0.020739333		
s^2	0.061163311	0.567706935	0.000208237	0.000252331	0.00013919		
s	0.247312173	0.753463294	0.014430424	0.015884933	0.011797888		
sn	0.036867122	0.112319676	0.002151161	0.002367986	0.001758725		

Table 6.2: *Experiment 2 Data Set*
Task Completion Time in Seconds

Replicate	*Dynamic*	*Static*	*Mouse*	*Keyboard*	*Joystick*	*RowTotal*	
1	254	214	102	102	107	779	
2	196	208	93	107	106	710	
3	152	193	91	107	106	649	
4	311	260	95	110	109	885	
5	122	253	96	106	102	679	
6	150	250	95	106	99	700	
7	146	183	91	105	105	630	
8	91	149	95	109	105	549	
9	89	142	93	109	101	534	
10	261	266	99	106	106	838	
11	145	211	98	104	103	661	
12	137	242	94	106	103	682	
13	73	83	90	104	103	453	
14	81	88	95	104	100	468	
15	78	88	93	103	102	464	
16	124	191	93	108	110	626	
17	124	199	98	105	101	627	
18	126	181	97	106	105	615	
19	128	182	92	108	106	616	
20	79	188	91	104	106	568	
21	93	155	90	106	107	551	
22	71	89	98	104	102	464	
23	74	83	97	107	103	464	
24	79	77	97	104	102	459	
25	121	153	97	106	102	579	
26	81	189	99	109	107	585	
27	85	183	98	107	101	574	
28	83	131	91	109	104	518	
29	76	137	92	106	107	518	
30	151	155	97	106	103	612	
31	95	142	95	104	103	539	
32	124	143	91	108	108	574	
33	89	149	96	109	102	545	
34	74	88	98	109	102	471	
35	70	80	98	109	100	457	
36	72	88	93	109	106	468	
37	136	184	98	104	107	629	
38	130	135	97	108	109	579	
39	125	142	93	102	102	564	
40	83	82	98	105	102	470	
41	74	81	99	108	106	468	
42	89	124	94	106	106	519	
43	75	89	95	108	106	473	
44	82	129	98	109	102	520	
45	85	137	98	109	102	531	
Sx	5184	6916	4288	4790	4686	25864	Grand Total
n	45	45	45	45	45		
$\overline{x}$	115.2	153.6888889	95.28888889	106.4444444	104.1333333		
Sx^2	720864	1196078	408982	510070	488288		
$(\mathrm{Sx})^2$	26873856	47831056	18386944	22944100	21958596		
Sd^2	123667.2	133165.6444	383.2444444	201.1111111	319.2		
s^2	2810.618182	3026.491919	8.71010101	4.570707071	7.254545455		
s	53.01526367	55.01356123	2.951288026	2.137921203	2.693426341		
sn	7.903048894	8.200937506	0.439952043	0.318702476	0.401512293		

Chapter 7

Results and Analysis

Data was collected from the execution or operation phase of the experiment. Conclusions are made based on this data. Interpretation of the experimental data is needed to draw valid conclusions. Interpretation is carried out in two steps, namely using descriptive statistics and hypothesis testing [44].
We used ANalysis of Variance (ANOVA) instead of t-tests, for analysis and interpretation of our result data. This is because, if we have 3 treatments to compare (A, B, C) then we would need 3 separate t-tests (comparing A with B, A with C, and B with C). In our case we have five treatments so we would need 15 separate t-tests. This would be time-consuming but, more importantly, it would be inherently flawed because in each t-test we accept a 5% chance of our conclusion being wrong (when we test for $p = 0.05$). So, in 15 tests we would expect (by probability) that one test would give us a false result. ANalysis Of Variance (ANOVA) overcomes this problem by enabling us to detect significant differences between the treatments as a whole. We do a single test to see if there are differences between the means at our chosen probability level.
An important assumption underlies the Analysis of Variance: that all treatments have similar variance. If there are strong reasons to doubt this then the data might need to be transformed before the test can be done. In practice, there is a simple way to check for "homogeneity of variance" [10]. We deal with this as well in our work.

7.1 Descriptive Statistics

Tables 7.1 and 7.2 present statistics based on data collected from experiment 1 and experiment 2, respectively. It is a common assumption that a joystick is one of the most convenient input modalities for controlling mobile robot navigation. Because of this, target of the TeleGaze system [23] was to bring average task completion time equal to joystick time. In our results, for experiment 1 the average task completion time for the static interface is 160% of the joystick time while for the dynamic interface it is 119% of the joystick time. In this way the dynamic interface improves the performance of the system significantly, i.e. by 26% of the average static interface task completion time. For experiment 2, the average task completion time for the static interface is 148% of the joystick time, while for the dynamic interface it is 111% of the joystick time, so the dynamic interface improves the performance of the system by 25% of the average static interface task completion time. This improvement is due to the merger of the feedback region and the gaze-contingent region in the dynamic interface, which eleminates multiple dwell times that occur in the static interface design.

Another important observation is that in the case of both gaze-contingent interfaces, variability of data about the mean is very high, while for all conventional modalities (mouse, key-board and joystick) variability is very low. This is very well evident from Figures 7.1 and 7.2.

Table 7.1: *Statistics of Data Collected for Experiment-1*

Description	*Dynamic*	*Static*	*Mouse*	*Keyboard*	*Joystick*
Sx	445	596.5	373.03	372.52	372.79
n	150	150	150	150	150
$\overline{x}$	2.966666667	3.976666667	2.486866667	2.483466667	2.485266667
Sx^2	1329.28	2456.67	927.7069	925.1786	926.5033
$(\mathrm{Sx})^2$	198025	355812.25	139151.3809	138771.1504	138972.3841
Sd^2	9.113333333	84.58833333	0.031027333	0.037597333	0.020739333
s^2	0.061163311	0.567706935	0.000208237	0.000252331	0.00013919
s	0.247312173	0.753463294	0.014430424	0.015884933	0.011797888
sn	0.036867122	0.112319676	0.002151161	0.002367986	0.001758725
$\overline{x}\pm\mathrm{sn}$	2.97±0.04	3.98±0.11	2.48±0.002	2.48±0.002	2.49±0.002

Table 7.2: *Statistics of Data Collected for Experiment-2*

Description	*Dynamic*	*Static*	*Mouse*	*Keyboard*	*Joystick*
Sx	5184	6916	4288	4790	4686
n	45	45	45	45	45
$\overline{x}$	115.2	153.6888889	95.28888889	106.4444444	104.1333333
Sx^2	720864	1196078	408982	510070	488288
$(\mathrm{Sx})^2$	26873856	47831056	18386944	22944100	21958596
Sd^2	123667.2	133165.6444	383.2444444	201.1111111	319.2
s^2	2810.618182	3026.491919	8.71010101	4.570707071	7.254545455
s	53.01526367	55.01356123	2.951288026	2.137921203	2.693426341
sn	7.903048894	8.200937506	0.439952043	0.318702476	0.401512293
$\overline{x}\pm\mathrm{sn}$	115.2±7.9	153.69±8.20	95.29±0.43	106.44±0.32	104.13±0.4

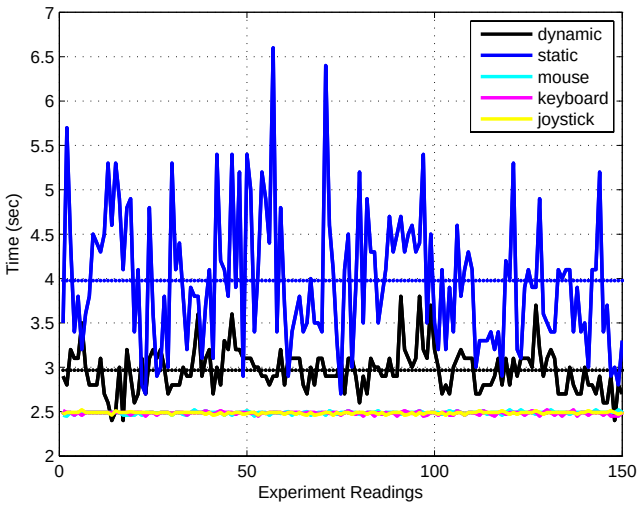

Figure 7.1: *Experiment-1 results.*

7.1.1 Hypothesis Testing

Tables 7.3 and 7.4 present the ANOVA statistics. For experiment 1, the calculated F value is 499.7643522, which exceeds F critical value of 2.38388541 needed in order to have a significant difference between treatments. The probability (p-value) that our calculated F value would be obtained by chance (random error) alone is very small (3.2501E-209), so we have a highly significant difference between treatments in our table 7.3. This holds for the experiment 2 as well. Based on the results for experiments 1 and 2, we can therefore reject the null hypothesis in favor of the claim that the dynamic interface takes less time compared to the static interface.

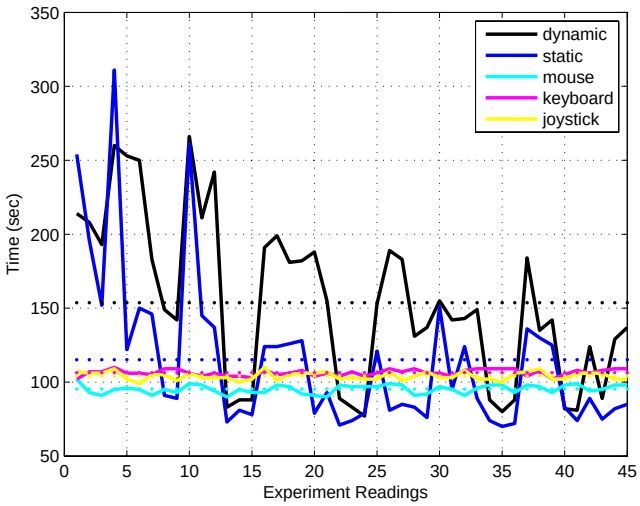

Figure 7.2: *Experiment-2 results.*

Table 7.3: *Summary of ANOVA test for Experiment-1*

Groups	*Count*	*Sum*	*Average*	*Variance*		
Dynamic	150	445	2.966666667	0.061163311		
Static	150	596.5	3.976666667	0.567706935		
Mouse	150	373.03	2.486866667	0.000208237		
Keyboard	150	372.52	2.483466667	0.000252331		
Joystick	150	372.79	2.485266667	0.00013919		
ANOVA						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	251.6693352	4	62.9173338	499.7643522	3.2501E-209	2.38388541
Within Groups	93.79103067	745	0.125894001			
Total	345.4603659	749				

Table 7.4: *Summary of ANOVA test for Experiment-2*

Groups	*Count*	*Sum*	*Average*	*Variance*		
Dynamic	45	5184	115.2	2810.618182		
Static	45	6916	153.6888889	3026.491919		
Mouse	45	4288	95.28888889	8.71010101		
Keyboard	45	4790	106.4444444	4.570707071		
Joystick	45	4686	104.1333333	7.254545455		
ANOVA						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	93450.06222	4	23362.51556	19.94189964	5.02577E-14	2.412682038
Within Groups	257736.4	220	1171.529091			
Total	351186.4622	224				

Chapter 8

Conclusion and Discussion

In this thesis we have introduced the concept of using a gaze contingent dynamic interface to remotely control a mobile robot. Then we have compared the performance of a dynamic interface with other input modalities for remote navigation control, with a special interest in comparison with TeleGaze [23] like static interface. The basic purpose of the dynamic interface design is to decrease the repeated dwell time of gaze-directed robot control resulting from switching between feedback and gaze contingent regions of the operator interface. A second purpose of the study was to facilitate the operator in such a way that she feels more comfortable with feedback monitoring tasks while sending navigation commands to the robot at the same time. Results of our pilot study show that the performance of our proposed dynamic interface is significantly better than that of the static interface. We have seen an improvement of about 25% in the performace of the dynamic interface compared to that of the static interface, in relation to the mean task completion time for control via a joystick.

Twelve out of fifteen participants reported that they felt more comfortable with using the dynamic interface. The rest of the participants voted in favour of the static interface. An interesting observation in this regard is that the participants who voted in favour of the static interface performed almost same with both interfaces i.e. their task completion time was almost same for both interfaces. All those participants who can drive or play computer

games completed the experiment task in less time while avoiding collisions more, compared to the rest of the participants.

In almost all trials each participant performed better in every subsequent trail i.e. she took less task completion time.

Another interesting observation can be made regarding training of the participants. In initial trails we handled each participant alone at the experiment site and she learned from her own experience. But later in the experiments we worked with groups of three or four participants on the experiment site at the same time. In this latter situation, when each participant was performing the experiment, the remaining were watching her activities. All such participants who watched others performed comparatively better on their turn and took smaller task completion times. This phenomenon can be seen in the graphical results for experiment 2. It is evident from the figure 7.2 that the task completion times for the initial 12 trials are very high and then we can see more consistent results for the rest of the trials. The region after 12 trials is the region where participants were present in groups.

Limitations of Gaze Interaction: The experiment observations shown on figures 7.1 and 7.2 show hight variation in the mean task completion time in the cases of both the static and the dynamic interfaces. This is due to rapid movements of eyes away from areas of interest falling within gaze contingent regions and consequent use of multiple dwell times. These movements are due to movement in areas of peripheral vision and eyes being consequently attracted towards those areas of peripheral vision. We have observed that this distraction can be controlled by training but cannot be eliminated completely. The effect of training is described earlier in conclusion text.

It was also observed during the experiments that it is hard to fixate in comparatively smaller areas. If some object is small enough that it is smaller than the area of high-acuity vision, then it subtends an angle of less than one degree from a normal viewing distance (as described in chapter 2). All such objects of this smaller size are difficult to fixate in. This is why it is hard to control standard graphical user interfaces using eye gaze as a pointing

method. Relatively larger regions in the interface are needed.

Dwell time is a substitute technique for mouse clicks but we know that in standard graphical user interfaces some objects need double clicks to perform associated actions. This is another limitation of eye gaze interaction with dwell time as a selection mechanism using it for standard graphical user interfaces.

8.1 Answers to Research Questions

In this section we present brief answers to the research questions provided by this thesis work

8.1.1 Research Question-1

How can different components, an Eye Gaze Tracker, a Tele-operation Station and a Mobile Robot, be integrated with each other to work as a whole system?

After careful study and analysis of technical documentation provided with different software and hardware components to be integrated we concluded the following:

- The Tobii Eye Tracker SDK provides ATL-COM (Active Template Library-Component Object Model) objects to provide real time gaze data in the form of x, y coordinates of user gaze points on the screen at a data rate of 60Hz.
- The XBee Wireless Communication Module provides a serial link to send the commands to the Arduino board on the Spinosaurus robot. C++ is the most efficient language to drive the serial link from the application system.
- OpenCV is an open source image processing tool from Intel. It is a C++ based tool. It can be used to grab the video stream sent wirelessly

from the camera mounted on the Spinosaurus and then to augment this stream with our new innovative interface elements.

This process of technical documentation investigation identifies data structures, interfaces and choice of language required to successfully communicate among diverse components. Based on this information software architecture of a working prototype is presented. This is discussed in length in chapter 5.

8.1.2 Research Question-2

How can we improve the performance of gaze interaction by making different arrangements of the active regions of the operator interface?

Based on gaze behaviour of the users presented in section 5.2 a new dynamic interface is developed as a part of the prototype. For empirical evaluation, in our specific case we have designed two experiments to collect evidence regarding whether our developed solution improves the performance of the system compared to existing solutions by other researchers in this area. Results show that this new dynamic interface performed quite well as compared to the static interface. Chapters 6 and 7 tell the whole story.

8.1.3 Research Question-3

What are the limitations of gaze interaction, if any, and the reasons for these limitations?

Based on the analysis of evidence collected from empirical evaluation in research question 2, we discuss limitations of gaze interaction. Details are presented in previous section of this chapter.

8.2 Future Work

In the current scenario each gaze contingent region sends a single command to the robot. There is very little variation available in different activities. For example, we can move left or right with a fixed angle or forward or backwards with a single speed. The dynamic interface has a quite big area. In a future variant of the interface this area could be used to bring variation in the degree of motion specified in commands. For example, the upper half area of the LEFT gaze contingent region can be used to turn at different angles by using different areas within the region, or different areas within the FORWARD region could be used to move with different speeds. This may be explored in future work. Performance issues may arise when we try to achieve this customization in speed levels and turning angle. Actual implementation may reveal the real results.

Bibliography

[1] Arduino. *Arduino Board Duemilanove*, page [online]. available: http://arduino.cc/en/Main/ArduinoBoardDuemilanove, 2010.

[2] Arduino. *Arduino Xbee-ZB-Module*, pages [online]. available: http://www.digi.com/products/wireless/zigbee–mesh/xbee–zb–module.jsp#overview, 2010.

[3] R. Atienza and A. Zelinsky. Active gaze tracking for human-robot interaction. *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces,IEEE Computer Society*, page 261, 2002.

[4] R. Barea, L. Boquete, M. Mazo, and E. Lopez. System for assisted mobility using eye movements based on lectrooculography. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10:209–218, 2002.

[5] W. Barfield and T. A. Furness. *Virtual Environments and Advanced Interface Design*. Oxford University Press, 2005.

[6] M. A. Bhuiyan, V. Ampornaramveth, S. Muto, and H. Ueno. On tracking of eye for human-robot interface. *International Journal of Robotics and Automation*, 19:42–54, 2004.

[7] R. A. Bolt. Eyes at the interface. *Human Factors in Computer Systems Gaithersburg,Maryland:ACM*, 1982.

[8] G. Bradski and A. Kaehler. *Learning OpenCV*. O Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2008.

[9] John W Creswell. *Research design : qualitative, quantitative, and mixed methods approaches*. Thousand Oaks : Sage, cop, 2003.

[10] J. Deacon. *Analysis of Variance*, page [online]. available: http://www.biology.ed.ac.uk/research/groups/jdeacon/statistics/tress6.html, 2010.

[11] A. T. Duchowski. *Eye Tracking Methodology: Theory and Practice*. Springer, New York, 2003.

[12] G. Dudek and M. Jenkin. *Computational principles of mobile robotics*. Cambridge University Press, 1999.

[13] Mark B. Friedman. Eyetracker communication system. *Proceedings of the Annual Symposium on Computer Application in Medical Care*, pages 895–896, 1983.

[14] D. C. Gordana. Constructive research and info-computational knowledge generation. *Studies in Computational Intelligence, Model-Based Reasoning in Science and Technology*, 314:359–380, 2010.

[15] H. K. Heidi and Simon P. Levine. Learning and performance of able-bodied individuals using scanning systems with and without word prediction. *Assistive Technology: The Official Journal of RESNA*, 6:42–53, 1994.

[16] C. Hofmeister, R. Nord, and D. Soni. *Applied Software Architecture*. Pearson Education North Asia Limited and Publishing House of Electronic Industry, 2003.

[17] R. Hyotylainen. Practical interests in theoretical considerations. constructive methods in the study of the implementation of informative systems. *Finland*, 2006.

[18] P. Isokoski. *Manual Text Input: Experiments, Models and System*. University of Tempere Finland, 2004.

[19] N. Jyrki. *On constructive research in computer science.* University of Tempere Finland, 2007.

[20] M. Kumar, J. Klingner, R. Puranik, T. Winograd, and A. Paepcke. Improving the accuracy of gaze input for interaction. *Proceedings of the 2008 symposium on Eye tracking research applications Savannah, Georgia:ACM*, pages 65–68, 2008.

[21] C. Lankford. Effective eye-gaze input into windows. *Proceedings of the 2000 symposium on Eye tracking research applications, Palm Beach Gardens, Florida, United States*, pages 23–27, 2000.

[22] C. Lankford. Effective eye-gaze input into windows. *Proceedings of the 2000 symposium on Eye tracking research applications Palm Beach Gardens, Florida,United States: ACM*, pages 23–27, 2000.

[23] H. Latif, N. Sherkat, and A. Lotfi. Telegaze: Teleoperation through eye gaze. *Cybernetic Intelligent Systems,2008. CIS 2008.7th IEEE International Conference*, pages 1–6, 2008.

[24] H. O. Latif, N. Sherkat, and A. Lotfi. Telegaze: Teleoperation through eye gaze. *Cybernetic Intelligent Systems, 2008. CIS 2008. 7th IEEE International Conference on*, pages 1–6, 2008.

[25] J. L. Levine. An eye-controlled computer. *IBM TJ Watson Research-Center: Yorktown Heights,New York*, 1981.

[26] C. S. Lin, C. W. Ho, W. C. Chen, C. C. Chiu, and M. S. Yeh. Powered wheelchair controlled by eye-tracking system. *Optica Applicata*, 36:401–12, 2006.

[27] K. Lukka. The constructive research approach. *Publications of the Turku School of Economics andBusiness Administration, Series B 1*, pages 83–101, 2003.

[28] M. Masood. *Awaz-e-Dost.* Sang-e-Meel Publishers Lahore Pakistan, 1983.

[29] Y. Mitsuho and F. Tadahiko. Eye word processor (ewp) and peripheral controller for the als patient. *IEE Proceedings A: Physical Science. Measurement and Instrumentation. Management and Education. Reviews*, 134:328–330, 1987.

[30] D. P. Noonan, G. P. Mylonas, A. Darzi, and G. Z. Yang. Gaze contingent articulated robot control for robot assisted minimally invasive surgery. *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008.*, 6:1186–1191, 2008.

[31] Dan R. Olsen and Michael A. Goodrich. Metrics for evaluating human-robot interactions. *Performance Metrics for intelligent Systems (PerMIS 2003)*, 2003.

[32] S. P. Parikh, R. Rahul, S. H. Jung, K. Vijay, J. P. Ostrowski, and C. J. Taylor. Human robot interaction and usability studies for a smart wheelchair. *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, 4:3206–3211, 2003.

[33] J. K. Robert and Jacob. You look at is what you get: Eye movement-based interaction techniques. *CHZ 90: ACM Conference on Human Factors in Computing Systems: Addison-Wesley/ACM Press*, 1990.

[34] A. J. R. Siddique. *A Vision and Differential Steering System for a Mobile Robot Platform.* Blekinge Institute of Technology Karlskrona, 2010.

[35] SmartboxAssistiveTechnology. *Software*, page [Online]. Available:http://www.smartboxat.com/software.html, 2010.

[36] M. Tall, A. Alapetite, D. W. Hansen, J. S. Agustin, E. Mllenbach, and H. H. Skovsgaard. Gazecontrolled driving. *27th International Conference Extended Abstracts on Human Factors in Computing Systems CHI 2009, April 4, 2009 - April 9, 2009, Boston, MA, United states: Association for Computing Machinery*, pages 4387–4392, 2009.

[37] Tobii. *Tobii_ProductDescription_TXSeriesEyeTrackers_230610_USeng*. Tobii, 2010.

[38] TobiiTechnologiesAB. *Tobii T60 and T120 Eye Trackers*, page [Online] Available: http://www.tobii.com/scientific_research/products_services/eye_tracking_hardware/tobii_t60_t120_eye_trackers.aspx, 2010.

[39] Trex. *Pololu TReX Dual Motor Controller*, page [online]. available: http://www.pololu.com/catalog/product/777, 2010.

[40] K. M. Tsui, D. J. Feil-Seifer, M. J. Matari, and H. A. Yanco. Performance evaluation methods for assistive robotic technology. *Performance Evaluation and Benchmarking of Intelligent Systems*, 2009.

[41] N. J. Wade and B. W. Tatler. *The Moving Tablet of the Eye: The Origins of Modern Eye Movement Research*. Oxford University Press, 2005.

[42] C. Ware and H. H. Mikaelian. An evaluation of an eye tracker as a device for computer input. *CHI+GI:ACM Conference on Human Factors in Computing Systems and Graphics Interace.Toronto*, 1987.

[43] Wikipedia. *Wikimedia Foundation, Inc.*, page [online]. available: http://en.wikipedia.org/wiki/Constructive_research, 2010.

[44] Claes Wohlin. *Experimentation in software engineering : an introduction*. Boston : Kluwer, cop, 2000.

[45] XBee. *XBee Module*, page [online]. available: http://www.coolcomponents.co.uk/catalog/product_info.php?cPath=25_64&products_id=330, 2010.

[46] XBee. *XBee Explorer Module*, pages [online]. available: http://www.trossenrobotics.com/store/p/5828–XBee–Explorer–USB.aspx, 2010.

[47] D. H. Yoo, J. H. Kim, D. H. Kim, and M. J. Chung. A human-robot interface using vision-based eye gaze estimation system. *IROS 2002: IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 196–201, 2002.

[48] S. Zhai. What is in the eyes for attentive input. *Communications of the ACM*, 46(3), 2003.

[49] S. Zhai, C. Morimoto, and S. Ihde. Manual and gaze input cascaded (magic) pointing. *Proceedings of the SIGCHI conference on Human factors in computing systems:the CHI is the limit Pittsburgh, Pennsylvania,United States: ACM*, pages 246–253, 1999.

Appendix A

The Spinosaurus Mobile Robot: System Description

The Spinosaurus is a small mobile robot that has been developed as a platform for research in mobile robot control and human/robot interaction. The first version of the Spinosaurus (1.0) was completed in April 2009. This version was a pure radio controlled system, using FM analog control from a conventional hobby RC controller to steer and drive the robot. A video camera mounted on the steerable turret of the robot sent an image to a receiver that was connected to a computer equipped with a Tobii T60 eye-tracking system. This configuration was used in a simple eye-tracking study to investigate human vision while remotely controlling the robot in a task to search for objects in a maze, using only the view from the robot camera as a guide. In 2010 the Spinosaurus has been upgraded to version 2.0 that includes:

- removal of the analog FM radio control system
- addition of a bi-directional data link based upon Xbee wireless data transceivers
- the addition of 2 x on-board Arduino boards for sensor data acquisition, telecommunications handling and motion control

The data link can be used to control the robot from an external computer.

A.1 Research Topics being Investigated

Research topics being investigated using this system includes:

1. Gaze-Directed Robot Control. An eye gaze tracking system detects the direction in which a user is looking. Gaze tracking is used for many purposes, including studying visual attention (e.g. on web pages, printed material, computer games) and providing hands-free control to allow those who have limited or no use of their hands to use word processors, email and the web. Gaze tracking can also be used to control a system such as a mobile robot. This research involves developing an integrated system that uses gaze data from a Tobii T60 eye gaze tracking system to control the Spinosaurus 2.0 robot. T60 eye-tracker has infrared diode arrays at the base of the screen and calculates the direction of gaze of a computer user based upon the distortions of the diode array patterns as reflected on the surface of the eye and detected by a camera embedded in the eye-tracker screen. Images from the video camera on the Spinosaurus robot are transmitted from the robot to a computer, and are then displayed on the eye-tracker screen. Robot control software on the operator computer can receive (x, y) gaze point data from the eye-tracker in real time via an application program interface (API) to the eye-tracker server. The control software can then use gaze points as interaction data similar to the (x, y) data provided by a mouse for controlling the operation of the robot.

2. Vision Processing for Autonomous Robot Control. Processing power on board the Spinosaurus robot is very limited. However, video images can be transmitted to an external computer and operational commands can be transmitted from the computer back to the robot. Hence it is possible to conduct computationally intensive vision processing for robot control by using a more powerful external computer for vision processing tasks. This research is currently investigating autonomous object detection and navigation using this system configuration.

A.2 Functional Architecture

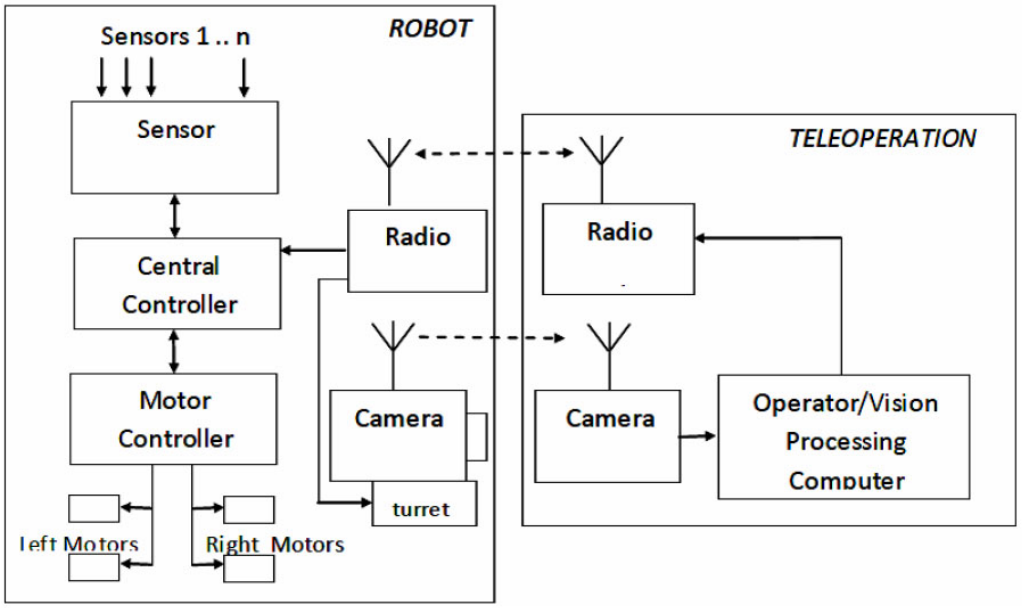

Figure A.1: *Robot Architecture Diagram*

A.3 Robot Parts List

- Lynxmotion A4WD mobile robot chassis
  (http://www.lynxmotion.com/Category.aspx?CategoryID=111)
- Pololu TReX DMC01 Dual Motor Controller
  (http://www.pololu.com/catalog/product/777)
- Arduino Diecimila Controller Board x 2
  (http://arduino.cc/en/Main/ArduinoBoardDiecimila)
- Devantech SRF08 Ultrasonic Rangefinder
  (http://www.robot-electronics.co.uk/acatalog/Ultrasonic_Rangers.html)
- SIR-01 Sharp GP2D12 IR sensor x 2
  (http://www.lynxmotion.com/images/data/gp2d12.pdf)

- Radio Transceivers: XBee 2mW Module with Whipantenna (http://www.coolcomponents.co.uk/catalog/product_info.php?cPath=25_64 &products_id=330)
- Radio Interface on Robot: xBee Shield for Arduino (http://www.coolcomponents.co.uk/catalog/product_info.php?cPath=50 &products_id=116)
- Radio Interface on Computer: xBee Explorer USB (http://www.coolcomponents.co.uk/catalog/product_info.php?cPath=25_64 &products_id=243)

A.4 Bibliography

www.tobii.se
http://www.ijcas.org/admin/paper/files/7608.pdf
http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=01307158
http://users.rsise.anu.edu.au/r̃sl/rsl_papers/atienzar_icmi2002.pdf
http://www.isd.mel.nist.gov/research_areas/research_engineering/Performance _Metrics/PerMIS_2003/Proceedings/Olsen.pdf
http://www.rose-hulman.edu/b̃erry123/pub_files/AreaPaper.pdf
http://gazeinteraction.blogspot.com/2008/11/eye-movement-control-of-remote-robot.html
http://www.irc.atr.jp/ỹone/research/pdf_folder/2008cogain.pdf
http://ieeexplore.ieee.org/ielx5/10831/34146/01626572.pdf?tp=

Appendix B

Application source code

```
...

float x, y, distance;
CString str;
time_t fixationTime;
long int currentTime;
currentTime = GetTickCount();

fixationTime = time(NULL);
showCursorCounter++;
gazeDataCount++;

if (gazeData->validity_lefteye == 0 && gazeData->validity_righteye == 0)
{
// Let the x, y and distance be the right and left eye average
x = (gazeData->x_gazepos_lefteye + gazeData->x_gazepos_righteye) / 2;
y = (gazeData->y_gazepos_lefteye + gazeData->y_gazepos_righteye) / 2;
distance = (gazeData->distance_lefteye + gazeData->distance_righteye) / 2;

// Set position, size and color of gaze form
int screenWidth  = GetSystemMetrics(SM_CXSCREEN);
int screenHeight = GetSystemMetrics(SM_CYSCREEN);
if(showCursorCounter \% 5 == 0){
m_pTrackedWindow->SetWindowPos(HWND_TOP, (int)(x * screenWidth) - 10,
(int)(y * screenHeight) - 10, 20, 20, SWP_SHOWWINDOW);
m_pTrackedWindow->SetBackgroundColor(RGB((int)distance \% 255, 255 -
((int)distance \% 255), 50));
m_pTrackedWindow->ShowXYPosition(x, y);

// get string pointer and show text window
ptrDirection = &direction;
m_pTrackedWindow->SetWindowTextA(ptrDirection);
}

// get global (x, y) coordinates
xValue = x*screenWidth;
yValue = y*screenHeight;

// collect gaze data for use in user gaze behavior analysis
gazeDataFile.open("gazedata.txt", fstream::in | fstream::out | fstream::app);
gazeDataFile << "x=\t";
gazeDataFile << (int)(x*screenWidth);
gazeDataFile << "\t";
gazeDataFile << "y=\t";
gazeDataFile << (int)(y*screenHeight);
gazeDataFile << "\t";
gazeDataFile << "time=\t";
gazeDataFile << fixationTime;
gazeDataFile << "\t";
gazeDataFile << "milliSecs=\t";
gazeDataFile << currentTime;
gazeDataFile << "\t";
gazeDataFile << "timeDiff=\t";
gazeDataFile << currentTime - dwellTime;

if(!firstTimeFlag){
totalMillisecondsNow = totalMillisecondsNow + (currentTime - dwellTime);
}
gazeDataFile << "\t";
gazeDataFile << "mSecElapsed=\t";
gazeDataFile << totalMillisecondsNow;
gazeDataFile << "\t";
gazeDataFile << "frameCount=\t";
gazeDataFile << frameCount;
gazeDataFile << "\t";
gazeDataFile << "dwellTime=\t";
gazeDataFile << dwellTime;
gazeDataFile << "\t";
gazeDataFile << "gazeDataCount=\t";
gazeDataFile << gazeDataCount;

// identify active regions and set direction or rotation angle later to be used
// for command preparation
if((xValue >= F1x && yValue >= F1y) && (xValue <= RECT4x && yValue <= RECT4y))
{
gazeDataFile << "\tintendedDirec=\t";
mainDialog->SetDirection('F');
gazeDataFile << mainDialog->GetDirection();
gazeDataFile << "\n";
}
```

```
else if(((xValue >= RECT1x && yValue >= RECT1y) && (xValue <= B4x && yValue <= B4y)))
{
gazeDataFile << "\tintendedDirec=\t";
mainDialog->SetDirection('B');
gazeDataFile << mainDialog->GetDirection();
gazeDataFile << "\n";
}
else if(((xValue >= RECT1x && yValue >= RECT1y) && (xValue <= R4x && yValue <= R4y)))
{
gazeDataFile << "\tintendedDirec=\t";
mainDialog->SetDirection('R');
gazeDataFile << mainDialog->GetDirection();
gazeDataFile << "\n";
}
else if((xValue >= L1x && yValue >= L1y) && (xValue <= RECT4x && yValue <= RECT4y))
{
gazeDataFile << "\tintendedDirec=\t";
mainDialog->SetDirection('L');
gazeDataFile << mainDialog->GetDirection();
gazeDataFile << "\n";
}
else
{
gazeDataFile << "\tintendedDirec=\t";
mainDialog->SetDirection('A');
gazeDataFile << mainDialog->GetDirection();
gazeDataFile << "\n";
}


gazeDataFile.close();
dwellTime = currentTime;
firstTimeFlag = 0;
}
return S_OK;
}

// function for capturing video stream from mobile robot camera and then
// superimposing gaze contingent interface on that stream.
LRESULT CTetClientSink::ShowFeedBackStream()
{
CvPoint pt1, pt2,ptBack1, ptBack2, ptLeft1, ptLeft2, ptRight1, ptRight2,
forwardDynamicPoint1, forwardDynamicPoint2, ptStopForward1, ptStopForward2, ptStopBackward1,
ptStopBackward2, backwardDynamicPoint1, backwardDynamicPoint2, rightDynamicPoint1,
rightDynamicPoint2, leftDynamicPoint1, leftDynamicPoint2, forwardLinePoint1, forwardLinePoint2,
forwardLinePoint3, forwardLinePoint4, backwardLinePoint1, backwardLinePoint2, backwardLinePoint3,
backwardLinePoint4, rightLinePoint1, rightLinePoint2, rightLinePoint3, rightLinePoint4,
leftLinePoint1, leftLinePoint2, leftLinePoint3, leftLinePoint4;
CvFont font, fontSmall;
double hScale = 1.0;
double vScale = 1.0;
double hScaleSmall = 0.5;
double vScaleSmall = 0.5;
int lineWidth = 1;

// set font size and type to be displayed in interface
cvInitFont( &font, CV_FONT_HERSHEY_DUPLEX | CV_FONT_ITALIC,
              hScale, vScale, 0, lineWidth );
cvInitFont( &fontSmall, CV_FONT_HERSHEY_DUPLEX | CV_FONT_ITALIC,
              hScaleSmall, vScaleSmall, 0, lineWidth );

// coordinates for gaze contingent regions
pt1.x = 60;
pt1.y = 0;
pt2.x = 580;
pt2.y = 50;
ptStopForward1.x = 60;
ptStopForward1.y = 50;
ptStopForward2.x = 580;
ptStopForward2.y = 120;
ptBack1.x = 60;
ptBack1.y = 430;
ptBack2.x = 580;
ptBack2.y = 480;
ptStopBackward1.x = 60;
ptStopBackward1.y = 360;
ptStopBackward2.x = 580;
ptStopBackward2.y = 430;
```

```
ptLeft1.x = 0;
ptLeft1.y = 60;
ptLeft2.x = 50;
ptLeft2.y = 350; //420-70 for stop active region
ptRight1.x = 590;
ptRight1.y = 60;
ptRight2.x = 640;
ptRight2.y = 350; //420-70 for stop active region
forwardDynamicPoint1.x = 60;
forwardDynamicPoint1.y = 0;
forwardDynamicPoint2.x = 580;
forwardDynamicPoint2.y = 350; //420-70 for stop active region
forwardLinePoint1.x = 320;
forwardLinePoint1.y = 0;
forwardLinePoint2.x = 320;
forwardLinePoint2.y = 420;
forwardLinePoint3.x = 60;
forwardLinePoint3.y = 210;
forwardLinePoint4.x = 580;
forwardLinePoint4.y = 210;
backwardDynamicPoint1.x = 60;
backwardDynamicPoint1.y = 130;
backwardDynamicPoint2.x = 580;
backwardDynamicPoint2.y = 480;
backwardLinePoint1.x = 320;
backwardLinePoint1.y = 480;
backwardLinePoint2.x = 320;
backwardLinePoint2.y = 60;
backwardLinePoint3.x = 60;
backwardLinePoint3.y = 270;
backwardLinePoint4.x = 580;
backwardLinePoint4.y = 270;
rightDynamicPoint1.x = 60;
rightDynamicPoint1.y = 60;
rightDynamicPoint2.x = 640;
rightDynamicPoint2.y = 350; //420-70 for stop active region
rightLinePoint1.x = 60;
rightLinePoint1.y = 240;
rightLinePoint2.x = 640;
rightLinePoint2.y = 240;
rightLinePoint3.x = 350;
rightLinePoint3.y = 60;
rightLinePoint4.x = 350;
rightLinePoint4.y = 420;
leftDynamicPoint1.x = 0;
leftDynamicPoint1.y = 60;
leftDynamicPoint2.x = 580;
leftDynamicPoint2.y = 350; //420-70 for stop active region
leftLinePoint1.x = 0;
leftLinePoint1.y = 240;
leftLinePoint2.x = 580;
leftLinePoint2.y = 240;
leftLinePoint3.x = 290;
leftLinePoint3.y = 60;
leftLinePoint4.x = 290;
leftLinePoint4.y = 420;

// capture video stream from mobile robot camera
CvCapture* capture = cvCaptureFromCAM( CV_CAP_ANY );

   if( !capture ) {
     fprintf( stderr, "ERROR: capture is NULL \n" );
     getchar();
     return -1;
   }

// Create a window in which the captured images will be presented
cvNamedWindow( "Control Interface", CV_WINDOW_AUTOSIZE );
// center of the screen 1280/4, 1024/4
cvMoveWindow("Control Interface", FrameWindowx, FrameWindowy);
```

```
  // Show the image captured from the camera in the window and repeat
while( 1 ) {
     // Get one frame
     IplImage* frame = cvQueryFrame( capture );

// set direction and rotation angle flag later to be used in commands
// and check dwell time before setting the direction and rotation angle flag
if ((xValue >= F1x && yValue >= F1y) && (xValue <= F4x && yValue <= F4y))||
(((xValue >= F1x && yValue >= F1y) && (xValue <= RECT4x && yValue <= RECT4y)) &&
dwellTimeFlagForward == 1)){
direction = 'F';
stopActiveRegionLocation = DOWN;
}
else if ((xValue >= B1x && yValue >= B1y) && (xValue <= B4x && yValue <= B4y)) ||
(((xValue >= RECT1x && yValue >= RECT1y) && (xValue <= B4x && yValue <= B4y)) &&
dwellTimeFlagBackward == 1)){
direction = 'B';
stopActiveRegionLocation = UP;
}
else if ((xValue >= R1x && yValue >= R1y) && (xValue <= R4x && yValue <= R4y)) ||
(((xValue >= RECT1x && yValue >= RECT1y) && (xValue <= R4x && yValue <= R4y)) &&
dwellTimeFlagRight == 1)){
direction = 'R';
stopActiveRegionLocation = DOWN;
}
else if ((xValue >= L1x && yValue >= L1y) && (xValue <= L4x && yValue <= L4y)) ||
(((xValue >= L1x && yValue >= L1y) && (xValue <= RECT4x && yValue <= RECT4y)) &&
dwellTimeFlagLeft == 1)){
direction = 'L';
stopActiveRegionLocation = DOWN;
}
else if (((xValue >= S1Bx && yValue >= S1By) && (xValue <= S4Bx && yValue <= S4By)) &&
stopActiveRegionLocation == DOWN) ||
(((xValue >= L1x && yValue >= L1y) && (xValue <= RECT4x && yValue <= RECT4y)) &&
dwellTimeFlagLeft == 1))*/{
direction = 'S';
stopActiveRegionLocation = UP;
}
else if(((xValue >= S1Fx && yValue >= S1Fy) && (xValue <= S4Fx && yValue <= S4Fy)) &&
stopActiveRegionLocation == UP){
direction = 'S';
stopActiveRegionLocation = DOWN;
}

// set the dwell time
if (direction == 'F'){
if((totalMillisecondsNow \% 400) > 0 && (totalMillisecondsNow \% 400) < 20){
dwellTimeFlagForward = 1;
dwellTimeFlagBackward = 0;
dwellTimeFlagRight = 0;
dwellTimeFlagLeft = 0;
}
}
else if (direction == 'B'){
if((totalMillisecondsNow \% 400) > 0 && (totalMillisecondsNow \% 400) < 20){
dwellTimeFlagForward = 0;
dwellTimeFlagBackward = 1;
dwellTimeFlagRight = 0;
dwellTimeFlagLeft = 0;
}
}
else if (direction == 'R'){
if((totalMillisecondsNow \% 400) > 0 && (totalMillisecondsNow \% 400) < 20){
dwellTimeFlagForward = 0;
dwellTimeFlagBackward = 0;
dwellTimeFlagRight = 1;
dwellTimeFlagLeft = 0;
}
}
```

```
else if (direction == 'L'){
if((totalMillisecondsNow \% 400) > 0 && (totalMillisecondsNow \% 400) < 20){
dwellTimeFlagForward = 0;
dwellTimeFlagBackward = 0;
dwellTimeFlagRight = 0;
dwellTimeFlagLeft = 1;
}
}
else {
if((totalMillisecondsNow \% 400) > 0 && (totalMillisecondsNow \% 400) < 20){
dwellTimeFlagForward = 0;
dwellTimeFlagBackward = 0;
dwellTimeFlagRight = 0;
dwellTimeFlagLeft = 0;
}
}

// update interface dynamically and send command to the robot

if(((xValue >= F1x && yValue >= F1y) && (xValue <= RECT4x && yValue <= RECT4y)) &&
direction == 'F'){

cvRectangle( frame, forwardDynamicPoint1, forwardDynamicPoint2, CV_RGB(0,255,0),
rectLineWidth, 8, 0 );
cvPutText( frame, "Move Forward", cvPoint( 200, 35 ), &font, cvScalar( 255, 0, 0 ) );
                cvRectangle( frame, ptBack1, ptBack2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvPutText( frame, "Move Backward", cvPoint( 200, 465 ), &font,
        cvScalar( 255, 0, 0 ) );
cvRectangle( frame, ptRight1, ptRight2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvRectangle( frame, ptLeft1, ptLeft2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvRectangle( frame, ptStopBackward1, ptStopBackward2,
CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvPutText( frame, "STOP", cvPoint( 260, 410 ), &font, cvScalar( 255, 0, 0 ) );
if(forwardFlag == 0){
SendCommand('w', '1', '0', '0');
forwardFlag = 1;
backwardFlag = 0;
leftFlag = 0;
rightFlag = 0;
stopFlag = 0;
}
}
else if(((xValue >= RECT1x && yValue >= RECT1y) && (xValue <= B4x && yValue <= B4y)) &&
direction == 'B'){

cvRectangle( frame, pt1, pt2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvPutText( frame, "Move Forward", cvPoint( 200, 35 ), &font, cvScalar( 255, 0, 0 ) );
cvRectangle( frame, backwardDynamicPoint1, backwardDynamicPoint2, CV_RGB(0,255,0), rectLineWidth, 8, 0 );
cvPutText( frame, "Move Backward", cvPoint( 200, 465 ), &font, cvScalar( 255, 0, 0 ) );
cvRectangle( frame, ptRight1, ptRight2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvRectangle( frame, ptLeft1, ptLeft2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvRectangle( frame, ptStopForward1, ptStopForward2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvPutText( frame, "STOP", cvPoint( 260, 90 ), &font, cvScalar( 255, 0, 0 ) );
if(backwardFlag == 0){
SendCommand('x', '1', '0', '0');
backwardFlag = 1;
forwardFlag = 0;
leftFlag = 0;
rightFlag = 0;
stopFlag = 0;
}
}
else if(((xValue >= RECT1x && yValue >= RECT1y) && (xValue <= R4x && yValue <= R4y)) &&
direction == 'R'){

cvRectangle( frame, pt1, pt2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvPutText( frame, "Move Forward", cvPoint( 200, 35 ), &font,
        cvScalar( 255, 0, 0 ) );
cvRectangle( frame, ptBack1, ptBack2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvPutText( frame, "Move Backward", cvPoint( 200, 465 ), &font, cvScalar( 255, 0, 0 ) );
cvRectangle( frame, rightDynamicPoint1, rightDynamicPoint2,
CV_RGB(0,255,0), rectLineWidth, 8, 0 );
cvRectangle( frame, ptLeft1, ptLeft2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvRectangle( frame, ptStopBackward1, ptStopBackward2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvPutText( frame, "STOP", cvPoint( 260, 410 ), &font, cvScalar( 255, 0, 0 ) );
```

```
if(rightFlag == 0 && frameCount % 10 == 0){
SendCommand('s', '0', '2', '0');
rightFlag = 0;
forwardFlag = 0;
backwardFlag = 0;
leftFlag = 0;
stopFlag = 0;
}
}
else if(((xValue >= L1x && yValue >= L1y) && (xValue <= RECT4x && yValue <= RECT4y)) &&
direction == 'L'){

cvRectangle( frame, pt1, pt2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvPutText( frame, "Move Forward", cvPoint( 200, 35 ), &font,
        cvScalar( 255, 0, 0 ) );
cvRectangle( frame, ptBack1, ptBack2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvPutText( frame, "Move Backward", cvPoint( 200, 465 ), &font, cvScalar( 255, 0, 0 ) );
cvRectangle( frame, ptRight1, ptRight2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvRectangle( frame, leftDynamicPoint1, leftDynamicPoint2, CV_RGB(0,255,0), rectLineWidth, 8, 0 );
cvRectangle( frame, ptStopBackward1, ptStopBackward2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvPutText( frame, "STOP", cvPoint( 260, 410 ), &font, cvScalar( 255, 0, 0 ) );

if(leftFlag == 0 && frameCount % 10 == 0){
SendCommand('a', '0', '2', '0');
leftFlag = 0;
forwardFlag = 0;
backwardFlag = 0;
rightFlag = 0;
stopFlag = 0;
gazeDataFile << "\t";
}
}
else if(((xValue >= S1Bx && yValue >= S1By) && (xValue <= S4Bx && yValue <= S4By)) &&
direction == 'S' && stopActiveRegionLocation == UP){

cvRectangle( frame, pt1, pt2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvPutText( frame, "Move Forward", cvPoint( 200, 35 ), &font,
        cvScalar( 255, 0, 0 ) );
cvRectangle( frame, ptBack1, ptBack2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvPutText( frame, "Move Backward", cvPoint( 200, 465 ), &font, cvScalar( 255, 0, 0 ) );
cvRectangle( frame, ptRight1, ptRight2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvRectangle( frame, ptLeft1, ptLeft2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvRectangle( frame, ptStopBackward1, ptStopBackward2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvPutText( frame, "STOP", cvPoint( 260, 410 ), &font, cvScalar( 255, 0, 0 ) );

if(leftFlag == 0 && frameCount % 10 == 0){
SendCommand('w', '0', '0', '0');
leftFlag = 0;
forwardFlag = 0;
backwardFlag = 0;
rightFlag = 0;
stopFlag = 0;
gazeDataFile << "\t";
}
}
else if(((xValue >= S1Fx && yValue >= S1Fy) && (xValue <= S4Fx && yValue <= S4Fy)) &&
direction == 'S' && stopActiveRegionLocation == DOWN){

cvRectangle( frame, pt1, pt2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvPutText( frame, "Move Forward", cvPoint( 200, 35 ), &font,
        cvScalar( 255, 0, 0 ) );
cvRectangle( frame, ptBack1, ptBack2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvPutText( frame, "Move Backward", cvPoint( 200, 465 ), &font, cvScalar( 255, 0, 0 ) );
cvRectangle( frame, ptRight1, ptRight2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvRectangle( frame, ptLeft1, ptLeft2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvRectangle( frame, ptStopForward1, ptStopForward2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvPutText( frame, "STOP", cvPoint( 260, 90 ), &font, cvScalar( 255, 0, 0 ) );

if(leftFlag == 0 && frameCount % 10 == 0){
SendCommand('w', '0', '0', '0');
leftFlag = 0;
forwardFlag = 0;
backwardFlag = 0;
rightFlag = 0;
stopFlag = 0;
}
}
else{
```

```
cvRectangle( frame, pt1, pt2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvPutText( frame, "Move Forward", cvPoint( 200, 35 ), &font,
        cvScalar( 255, 0, 0 ) );
cvRectangle( frame, ptBack1, ptBack2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvPutText( frame, "Move Backward", cvPoint( 200, 465 ), &font, cvScalar( 255, 0, 0 ) );
cvRectangle( frame, ptRight1, ptRight2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvRectangle( frame, ptLeft1, ptLeft2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvRectangle( frame, ptStopBackward1, ptStopBackward2, CV_RGB(255,0,0), rectLineWidth, 8, 0 );
cvPutText( frame, "STOP", cvPoint( 260, 410 ), &font, cvScalar( 255, 0, 0 ) );
leftFlag = 0;
forwardFlag = 0;
backwardFlag = 0;
rightFlag = 0;
stopFlag = 0;
}

// check if video frames are not available

    if( !frame ) {
      fprintf( stderr, "ERROR: frame is null...\n" );
      getchar();
      break;
    }

    cvShowImage( "Control Interface", frame );
    // Do not release the frame!

// press Esc key to stop the application

if( (cvWaitKey(10) & 255) == 27 ){
mainDialog->SetStreamOff();
break;
}
frameCount++;
}

  frameCount++;

  // Release the capture device housekeeping
  cvReleaseCapture( &capture );
  cvDestroyWindow( "Control Interface" );
  return 0;
}

// prepare and send command to the mobile robot
// serial communication module
void CTetClientSink::SendCommand(char char1, char char2, char char3, char char4)
{
// open port for I/O
HANDLE h = CreateFile(TEXT("COM3"),
GENERIC_READ|GENERIC_WRITE,
0,NULL,
OPEN_EXISTING,0,NULL);

if(h == INVALID_HANDLE_VALUE) {
PrintError("E012_Failed to open port");
} else {
// set timeouts
COMMTIMEOUTS cto = { 1, 100, 1000, 0, 0 };
DCB dcb;
if(!SetCommTimeouts(h,&cto))
PrintError("E013_SetCommTimeouts failed");

// set DCB
memset(&dcb,0,sizeof(dcb));
dcb.DCBlength = sizeof(dcb);
dcb.BaudRate = 9600;
dcb.fBinary = 1;
dcb.fDtrControl = DTR_CONTROL_ENABLE;
dcb.fRtsControl = RTS_CONTROL_ENABLE;
dcb.Parity = NOPARITY;
dcb.StopBits = ONESTOPBIT;
dcb.ByteSize = 8;

if(!SetCommState(h,&dcb))
PrintError("E014_SetCommState failed");
```

```
char buf[7];
DWORD read = 0;
DWORD write=1; // Number of bytes to write to serial port
// take decision for direction and angle based on parameters
if (direction == 'F' && stopActiveRegionLocation == DOWN){
buf[0] = char1; // Character value to write to serial port
WriteFile(h, buf, write, &write, NULL); // write is updated with the number of bytes written
buf[0] = char2; // Character value to write to serial port
WriteFile(h, buf, write, &write, NULL); // write is updated with the number of bytes written
}
else if (direction == 'B' && stopActiveRegionLocation == UP){
buf[0] = char1; // Character value to write to serial port
WriteFile(h, buf, write, &write, NULL); // write is updated with the number of bytes written
buf[0] = char2; // Character value to write to serial port
WriteFile(h, buf, write, &write, NULL); // write is updated with the number of bytes written
}
else if (direction == 'S'){
buf[0] = char1; // Character value to write to serial port
WriteFile(h, buf, write, &write, NULL); // write is updated with the number of bytes written
buf[0] = char2; // Character value to write to serial port
WriteFile(h, buf, write, &write, NULL); // write is updated with the number of bytes written
}
else if (direction == 'R' || direction == 'L'){
buf[0] = char1; // Character value to write to serial port
WriteFile(h, buf, write, &write, NULL); // write is updated with the number of bytes written
buf[0] = char2; // Character value to write to serial port
WriteFile(h, buf, write, &write, NULL); // write is updated with the number of bytes written
buf[0] = char3; // Character value to write to serial port
WriteFile(h, buf, write, &write, NULL); // write is updated with the number of bytes written
buf[0] = char4; // Character value to write to serial port
WriteFile(h, buf, write, &write, NULL); // write is updated with the number of bytes written
}
CloseHandle(h);
}
}
```

Index