SELECTING AND COMMANDING INDIVIDUAL ROBOTS IN A MULTI-ROBOT SYSTEM

by

Alex Couture-Beil
B.Sc., Vancouver Island University, 2007

a Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Spring 2010

APPROVAL

Name: Alex Couture-Beil

Degree: Master of Science

Title of Thesis: Selecting and Commanding Individual Robots in a Multi-Robot System

Examining Committee: Dr. Brian Funt
Chair

Dr. Richard Vaughan, Associate Professor
School of Computing Science
Senior Supervisor

Dr. Greg Mori, Assistant Professor
School of Computing Science
Supervisor

Dr. Arthur Kirkpatrick, Associate Professor
School of Computing Science
Examiner

Date Approved: March 18, 2010

Declaration of
Partial Copyright Licence

Abstract

In this thesis, we present a novel real-time computer vision-based system for facilitating interactions between a single human and a multi-robot system: a user first selects an individual robot from a group of robots, by simply looking at it, and then commands the selected robot with a motion-based gesture. We describe a novel multi-robot system that demonstrates the feasibility of using face contact and motion-based gestures as two non-verbal communication channels for human-robot interaction.

Robots first perform face detection using a well-known face detector. The resulting "score" of the detected face is used in a distributed leader election algorithm to estimate which robot the user is looking at. The selected robot then derives a set of motion features, based on blurred optical flow, which is extracted from a user-centric region. These motion cues are then used to discriminate between gestures (robot commands) using an efficient learned classifier.

Acknowledgments

This work would not have been possible without the direction, expertise and support of my supervisors Dr Richard Vaughan, and Dr Greg Mori. I would like to extend my thanks to members of the Autonomy lab for their assistance and encouragement, and in particular thank Jens Wawerla and Ash Charles for help with the Chatterbox robots. I would also like to thank Mark Bayazit for his help in producing the early stages of the gesture recognition software.

Contents

Approval ii

Abstract iii

Acknowledgments iv

Contents v

List of Tables viii

List of Figures ix

1 Introduction 1

1.1 Motivation 2

1.1.1 Face engagement 3

1.1.2 Motion-based gestures 3

1.2 Contributions 3

1.3 Outline 4

2 Background 6

2.1 Face Engagement 6

2.2 Gaze as an input device 7

2.3 Gaze and interactive robots 8

2.4 Robot selection and task delegation 12

2.5 Gesture based robot interaction 14

3 Robot selection **16**
3.1 Face detection . . . 18
3.1.1 Viola-Jones face detection . . . 18
3.1.2 Face score . . . 21
3.2 Leader election . . . 21
3.2.1 A continuous leader election . . . 24
3.2.2 Ring network . . . 27
3.3 Conclusion . . . 32

4 Motion-based gesture recognition **33**
4.1 Introduction . . . 33
4.2 Algorithm . . . 35
4.2.1 Motion features . . . 36
4.2.2 Face detection . . . 36
4.2.3 Classification . . . 37
4.3 Implementation . . . 38
4.3.1 CUDA classification optimization . . . 38
4.4 Results . . . 39
4.5 Retraining for use on a mobile robot . . . 42
4.6 Conclusion . . . 43

5 The robot: putting it together **44**
5.1 The physical robot . . . 44
5.2 Demonstration task . . . 46
5.3 The robot controller . . . 47
5.3.1 Localization . . . 49
5.3.2 Navigation . . . 55
5.3.3 Controller summary . . . 56
5.4 Conclusion . . . 56

6 Discussion **59**
6.1 Demonstration . . . 59
6.2 Robot selection . . . 60
6.3 Task designation . . . 62

6.4 Navigation 63

7 Conclusion **64**

7.1 Contributions 64

7.2 Future work 65

7.3 Final comments 66

A RPC methods **67**

A.1 Controller RPC definitions 67

A.2 Laptop RPC definitions 69

Bibliography **72**

List of Tables

3.1 JSON-encoded election message 31
3.2 JSON-RPC election message passing 31

4.1 Classifier real-time performance 40
4.2 Classified gestures confusion matrix 42

5.1 Responsibilities of the controller and laptop 48

List of Figures

1.1 Sample images of selecting and commanding a robot 4

2.1 Kismet 8
2.2 Robovie 9
2.3 PeopleBot with gaze 11

3.1 Commanding with face engagement (cartoon) 17
3.2 Robot's view of the user 18
3.3 Face detection features 19
3.4 Face detector 20
3.5 Frontal face detector 22
3.6 Ring-based leader election 26
3.7 Ring networks 28
3.8 Joining a ring 29

4.1 Sample gestures 34
4.2 Motion cues used to classify the gesture 35
4.3 Parallel classifier speed-up factor 40
4.4 Precision-recall results 41
4.5 Robot-specific gestures 43

5.1 iRobot Create 45
5.2 Chatterbox-build photos 45
5.3 Gumstix integration board 46
5.4 Create with custom sensor board and laptop 47
5.5 Demonstration task environment 48

5.6 Network timing diagram 50
5.7 Odometry correction 52
5.8 Fiducial marker used for localization 53
5.9 Sample of stationary robot pose estimates 54
5.10 Robot circular arc formation 56
5.11 Election and gesture recognition flowchart 58

6.1 Tie-breaking by getting closer 61

Chapter 1

Introduction

Over the past two decades, the mobile robotics field has seen an increase of research related to multiple robot systems and distributed intelligence [76]. Prior to this shift, roboticists focused on single robots. Having a robot autonomously navigate between different rooms was a challenge [72]. As robot designs materialized, the multi-robot field emerged; rather than studying interactions between a robot and its environment, attention shifted to interactions among teams of robots. Early work involving foraging tasks with relatively low communication explored emergent properties of the system [61]. As larger groups of robots were experimented with, researchers developed methods for autonomously allocating tasks between robots [30].

An example of the maturity of the multi-robot field is the annual RoboCup competition. The aim is to develop a team of autonomous soccer playing robots in order to further research and education in the robotics field [50]. The official goal (perhaps tongue-in-cheek) is to produce a team of robots capable of beating Brazil (or the most recent World Cup champion) in a standard FIFA[1] game of soccer.

As the momentum of the robotics community increased, in particular with behaviour based reactive robots, interactions between robots and humans increased. Robots no longer exist solely in science fiction, and are now turning up in our homes as children's toys or vacuum cleaners. These daily interactions are studied in the field of human-robot interaction (HRI). HRI is a relatively young field; one might argue it first appeared as Asimov's three laws of robotics [2]; however, the first HRI-dedicated scientific conference, the IEEE

[1]Fédération Internationale de Football Association (International Federation of Association Football)

International Symposium on Robot and Human Interactive Communication (RoMan), first appeared in 1992 and occurs annually [33].

HRI is a broad multidisciplinary field which obviously includes robotics, but also includes (and is not limited to) psychology, sociology, and philosophy. The primary objective is to study the *interactions* between humans and robots; or in some cases between humans and what the human perceives to be a robot. These robots are not necessarily autonomous; in some cases teleoperation[2] is used to imitate intelligent behaviours (e.g. [37]).

The work presented in this thesis presents a complete autonomous multi-robot system with a computer vision-based task delegation interface.

1.1 Motivation

While you might still be waiting for your personal jetpack, robots are no longer a thing of the future; they are here, and here to stay. The robotics industry has already moved to mass production (e.g. iRobot), and daily interactions with robots are increasing. As these interactions increase, the question of how to effectively *communicate* with robots becomes a valid question. Robots with a low level of autonomy will require a great deal of human instruction to complete basic tasks such as navigation; however, as robots achieve higher levels of autonomy our communication methods will change.

Robots are found in many different environments, performing many different tasks. For example, robots are often used in hazardous environments, which are too dangerous for humans, performing tasks like space exploration (e.g. Mars exploration rovers [90]), search and rescue (e.g. [15]), and demining (e.g. [81]). These robots are teleoperated from remote locations – no sensor-mediated human-robot interactions occur, since the humans and robots are not co-located. On the other hand, robots can also be found *in* our environment performing tasks such as delivering items in an hospital (e.g. [67]), entertaining us (e.g. [28]), aiding with robot-assisted therapy (e.g. [53]) and activity (e.g. [100]), or cleaning our home (e.g. iRobot Roomba). These are the robots we focus on in this thesis: robots that *share* the same environment as us.

These robots are embodied in the same environment as us, and will interact with us on a daily basis in a face to face manner. It is important that they are able to communicate

[2] commonly referred to as the Wizard of Oz technique

with us in some convenient manner, and therefore must be able to understand our various communication channels including non-verbal communication. Below, we will investigate the use of two of these non-verbal communication channels, face engagement and motion-based gestures, as an interface for a multi-robot system.

1.1.1 Face engagement

We use the term *Face engagement*, coined by Goffman [31], to refer to the process in which people use eye contact and facial gestures to interact with people. We believe that face engagement could be an effective non-verbal communication channel for human-robot interactions. There is limited work in HRI that focuses on communication between a human and multi-robot system; to our knowledge, no work has investigated the use of face engagement as a selection mechanism for multi-robot systems. In this thesis we will investigate the feasibility of using face engagement to select individual robots from a group or robots, by building and demonstrating a real system.

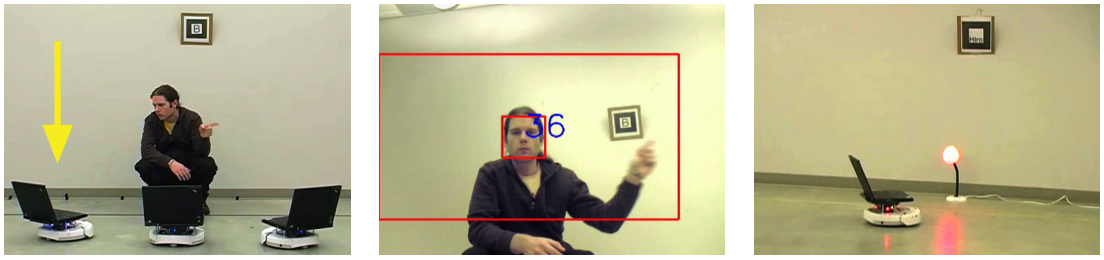
1.1.2 Motion-based gestures

Once an individual robot has been selected from the group of robots, motion-based gestures will be used to command the selected robot.

1.2 Contributions

In this thesis we address and contribute to the following problems:

- How to build a computer vision-based interface for an autonomous multi-robot system. We focus on the various components that make up such a system and describe how to effectively combine them together to produce a complete working system.
- How to select a particular robot from a group of robots with face engagement. We make a novel observation, that it is possible to use a frontal face detector to estimate which video camera a user is looking at by comparing the detected-face scores of each video-camera (which simultaneously captures the same person under similar lighting, but from different angles). Each robot then compares its score by means of a distributed leader election algorithm to guarantee only a single robot will ever be elected at a given time. This method is proposed as an alternative to performing eye gaze detection.



(a) A user selects an individual robot by looking at it, and assigns it a task by waving his hand.

(b) A user-centric region is identified; a learned classifier uses optical flow from the region to discriminate between gestures

(c) Robots travel to one of two zones as commanded by the user; colours are only used to illustrate different zones – robots use fiducial markers for localization

Figure 1.1: An example of selecting and commanding an individual robot from a group of robots.

- How to use motion-based gesture as a human robot interface to assign tasks to robots. We describe a real-time motion-based gesture recognition system based on machine learning techniques.

1.3 Outline

In this thesis, we will describe the various components required to produce a complete, working multi-robot system. To investigate the feasibility of our proposed interface, we will provide a demonstration task as a proof of concept. A three minute video [17] of our demonstration was published in the video track of the 5th annual ACM/IEEE International Conference on Human-Robot Interaction (2010), held in Osaka, Japan. Sample frames from our video, shown in Fig 1.1, illustrate a user selecting and commanding an individual robot from a group a robots. A follow-up paper [18] describing our system will appear in the seventh Canadian Conference on Computer and Robot Vision (CRV2010), to be held in Ottawa, Canada

The outline of this thesis is as follows:

Chapter 2: Background

First, we give an overview of background work describing the importance of eye gaze in social interactions. We then describe related works that make use of eye gaze in human-computer interactions and socially interactive robotics, as well as describe other methods

for robot selection and gesture-based human-robot interfaces.

Chapter 3: Robot selection

Next, we present the *robot selection problem*, and discuss our method for selecting an individual robot from a group of robots by using face engagement. Our implementation makes use of a distributed ring-based leader election algorithm over a wireless network to select the robot with the highest detected-face score.

Chapter 4: Motion-based gesture recognition

In chapter 4, we describe a *real-time system for gesture recognition*. Given a set of input frames, we derive a user-centric set of motion features based on smoothed optical flow estimates and face detection. An efficient classifier is learned to discriminate between different gestures, which will effectively be used to command robots. Experimental results demonstrate the speed and efficacy of our system.

Chapter 5: The robot: putting it together

A detailed description of the *multi-robot system* is described in chapter 5. The two computer vision components, described in the previous two chapters, are combined to build a functioning multi-robot system. A fiducial based localization system is used to navigate the described environment. A demonstration task, which is used to test the system, is presented.

Chapter 6: Discussion

A discussion of the observations of the demonstration task, performed by 7 participants, follows in chapter 6.

Chapter 7: Conclusion

Finally, we conclude the thesis and outline possible future work in chapter 7.

Chapter 2

Background

2.1 Face Engagement

Before two or more people can enter into a focused interaction, they must somehow mutually signal their mental focus and readiness. Eye contact and eye gaze plays an important role in initiating and regulating communication between people [48]. Goffman uses the term *face engagement* to describe the process in which people use eye contact, gaze and facial gestures to interact with or engage each other. Eye contact and gaze can be used to signal turns and roles throughout conversations. Face engagement is often used to signal one's attentiveness during exchanges of verbal statements; however, young children who have yet to master language use face engagement to signal their cognitive focus and visual attention [31].

The role of eye contact plays such an important role in the development of humans [51, 97] that the ability to detect eye contact is present at birth [23]. Newborns are particularly interested in faces with opened eyes rather than closed eyes [5]. Attentiveness to gaze direction quickly develops during a child's infancy; by 3–4 months, infants are more responsive to faces with direct gaze rather than averted gaze [24], by 10–11 months, infants develop gaze-following behaviour [12]. Those infants who showed gaze following behaviours at a younger age also showed a quicker vocabulary growth rate [13]; slower development of eye contact and gaze awareness however, can be an early indication of autism [106].

Eye contact and gaze play an important role as a non verbal communication channel in social interactions. Field experiments have shown that hitch-hikers who made eye contact were more likely to be offered a ride [89], library patrons were more likely to approach a

librarian when eye contact was made [82], and strangers on the street were more likely to answer a survey when confederates maintained eye contact throughout the request [35].

2.2 Gaze as an input device

From its origins in psychology, gaze tracking research can now be found in computing applications [20]. Eye gaze trackers range from intrusive systems which measure electrical voltage induced by a coil embedded in a contact lens [84] to non-invasive computer vision-based systems which never come into contact with the user [3]. Morimoto and Mimica provide an in-depth survey of gaze tracking techniques [66].

Gaze tracking systems have become tools in usability studies, where researchers are interested in recording the eye movements of humans as they interact with computer interfaces [32], navigate websites [9], perceive advertising in the Yellow Pages [55] and even fly a plane [93]. In addition to recording eye gaze for subsequent analysis, the human computer interaction (HCI) community has studied real-time eye gaze tracking systems as input devices [44]. One of the earlier interactive systems, presented by Jacob and Karn, uses gaze to select objects and display corresponding information which can be scrolled through by the eye. He presents a variation of the "Midas touch" problem[1]: how can one differentiate between looking at versus selecting an object since there's no way to click with eye gaze. He observes that the use of dwell time acts as a good "clicking" mechanism [43]. Another early system, developed by Starker and Bolt, present a virtual 3D world[2] with an interactive story that changes depending on which objects the user is looking at [91].

Gaze based interfaces are of particular interest for developing hands-free systems such as helmet tracking systems for fighter jet missile controllers where a pilots hands are occupied flying a jet [26], or for developing communication devices such as text-input systems for disabled users who cannot use their hands [57]. Gaze pointing devices, however, are inaccurate; Zhai et al. propose a hybrid cursor system where eye gaze indicates a general region to reposition the cursor to, and then a mouse is used to precisely select objects [107].

While face engagement is not directly an input device for video-teleconferencing, it is often unachievable between participants due to the different location of the video screen versus the video camera – which results in unnatural conversations. In order to simulate

[1] His problem does not produce any gold

[2] likely inspired by the French novel "Le Petit Prince" by Antoine de Saint-Exupry

natural face engagement, participants' eye gaze must be corrected [104].

2.3 Gaze and interactive robots

With the HRI community's interdisciplinary roots drawing upon fields like psychology and cognitive science, the importance of eye contact and gaze awareness, especially with humanoid robotics, is well known [54, 86, 68]. A trend in HRI is to give robots human characteristics; researchers argue that anthropomorphizing robots will lead to more natural interactions with humans. By exploiting human familiarity, implementing basic social cues in robots can help people develop social relationships with robots [21]. When people talk about these social robots they may linguistically anthropomorphize robots by using words like "love" or "hate" (e.g. the robot *loves* to recharge); however, when asked whether or not a robot could feel affection, or could have a certain mood, their judgements remain inanimate [29]. While creating a synthesized human [41] may lead to unrealistic prior expectations, some level of anthropomorphization of communication channels should be investigated [21], and in particular to this thesis, the use of eyes as a non-verbal communication channel for a human-robot interface is explored.

A highly whimsical responsive sociable robot is Kismet, shown in Fig 2.1, which was developed by Breazeal at MIT [11]. Kismet was designed to explore and study emotional behaviours in robotics. Inspired by infant social development, Kismet was designed to be sensitive to highly saturated and skin tone colours, faces, motion, the distance or size of the person or object, and voice intonation. Kismet, however, does not respond to or communicate with speech – but rather limits interactions to synthesized emotions. Nonetheless, the sounds, facial gestures, and gaze produced by Kismet is effective in communicating the emotional state and attentiveness of the robot.

One humanoid robot at the center of several papers is Robovie, which was developed by Ishiguro et al. at ATR [42]. Robovie, seen in Fig 2.2, has two controllable eyes, thus allowing the use of gaze as a communication channel [40].

In an experiment by Mutlu et al. gaze is used to regulate conversations between Robovie and two human participants [68]. The participants are assigned one of three roles which determine the amount of interaction with the robot: an *addressee* who the robot speaks to, a *bystander* whose presence is only acknowledged, or an *over-hearer* who witnesses the conversation from a distance. Three different participant configurations are explored in the

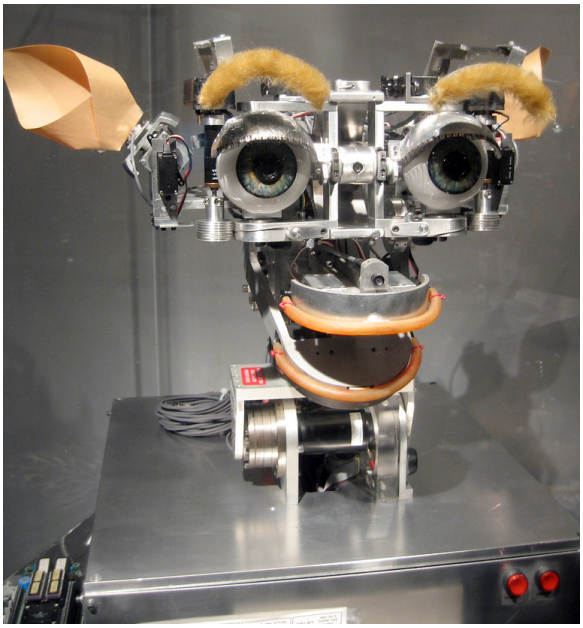


Figure 2.1: Kismet, developed by Breazeal, on display in the MIT museum. Picture by Nadya Peek; used by permission.

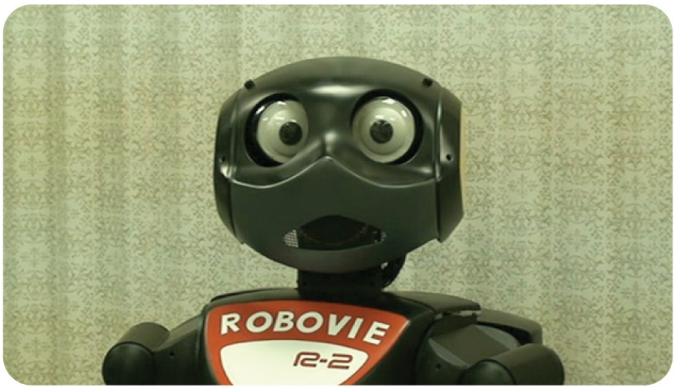


Figure 2.2: The humanoid robot Robovie, developed at ATR by Ishiguro et al. Picture by Bilge Mutlu; used by permission.

experiment:

1. **Two addressees**: Both participants are acknowledged during the greeting phase and throughout the entire conversation by the robot. The robot's attention is equally divided between the two addresses, and eye contact is used to yield speaking turns to a particular addressee.

2. **An addressee and a bystander**: During the greeting phase, the robot acknowledges both participants. The majority of the robot's attention throughout the conversation is focused on the addressee with quick less frequent glances towards the bystander.

3. **An addressee and an over-hearer**: The final configuration involves an addressee as usual and an over-hearer who is never acknowledged or greeted throughout the entire conversation.

From the subjective evaluation of the 72 participants, the authors found that participants who were assigned the over-hearer role (who were never acknowledged by the robot) liked the robot significantly less than the other participants. Eye gaze was also a factor in the "feeling of groupness" score, which addressees significantly rated a higher. In addition to increasing user experience ratings, the authors also found that gaze was an effective tool for yielding speaking turns and reinforcing conversation roles.

In an earlier experiment with Robovie, Imai et al. investigated the relationship between the robot's gaze and head position [39]. Eight participants were seated in a circle centered around the robot, whose head and gaze would try to follow whoever was speaking. The authors found that participants only noticed Robovie's gaze after noticing the head movement.

Besides yielding speaking roles and regulating conversation, gaze can also be used to establish *joint attention* between a speaker and addressee. Joint attention is the phenomenon where two or more people mutually focus on the same visual stimuli as alerted by one of the parties through a non-verbal channel such as eye gaze, head movement or pointing. The work of computational linguists Staudte and Crocker explores the effects of joint attention between a robot and a human participant. 48 participants (unfortunately) watched pre-recorded videos of a robot making a statement related to objects lined up in front of it, as seen in Fig 2.3. The robot would then gaze at a particular object to focus the joint attention to the relevant object; in some cases the robot purposely gazed at an irrelevant

object which lead to higher cognitive response time from the participants. The authors claim that implementing human-like gaze in the robot improves human comprehension of robot speech [92]. Similar work by Mutlu et al. investigate the role of nonverbal leakage, that is, seemingly unintentional cues containing information, in human-robot interactions [69].

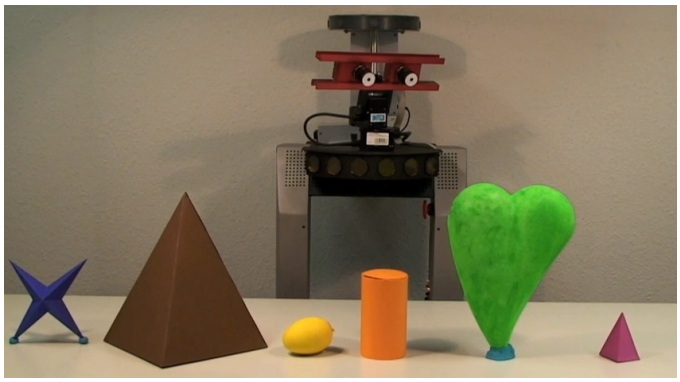


Figure 2.3: A modified MobileRobots PeopleBot, used by Staudte and Crocker, gazes at an object in order to establish joint attention with a human partner. Picture by Maria Staudte; used by permission.

So far we have only looked at human responses to a robot's gaze, Kuno et al. argue that joint attention must be perceived by both the human and robot. Their work emphasizes the need to synchronize speech and vision processing in order to build a useful human-robot interface [54]. Without vision, the robot would be unable to pick up on joint attention cues such as gaze or pointing. The authors present a prototype museum guide robot which is activated by eye contact. Rather than truly performing gaze detection of the pupils, the system relies on the detection of frontal faces. Once a face has been detected, a telephoto lens is used to capture a high quality image; the robot then estimates if the user is looking at it by detecting if the nostrils are centered between the eyes. Their "eye contact" detector, or perhaps what should be referred to as a "face engagement" detector, allows for interactions up to a distance of six meters. When a museum visitor makes eye contact or visually engages the robot, the robot approaches the user and offers to explain the exhibit.

Another eye gaze aware robot is presented as a "daily-partner robot" by Yonezawa et al. in [105]. The authors claim the robot will help satisfy the emotional needs of patients

that require nursing care – albeit in reality the robot in its current prototype appears to be little more than an augmented answering machine dressed in a teddy-bear outfit. The cross-modal perception technology behind it, nonetheless, is exciting. The robot performs both eye gaze tracking through a high resolution wide angle camera as well as speech recognition. To interact with the robot the user must first look at the robot to make eye contact and then utter a command; in this case eye contact is used to designate the robot as the addressee of the user. This is done to disambiguate situations where the user is talking *to* the robot versus someone else in the room. When the robot needs to alert the human user of a message, rather than potentially disturbing the user with an audible message when the user is preoccupied, the robot waits to make eye contact before communicating the message.

Literature on eye gaze or face engagement aware human-robot interfaces is limited. While some of the robots discussed here only respond when looked at, it is not absolutely clear how precise their gaze tracking system is or how well it would fare in multi robot situations. We believe that the system presented in this thesis which proposes the use of face engagement for the selection of a particular robot amongst a group of closely spaced robots is the first of its kind.

2.4 Robot selection and task delegation

There is little work on human-robot interfaces for multi-robot systems. However, as multi robot systems, such as the iRobot Swarmbots, become more common one becomes aware that not all interaction methods for a single robot will be able to transfer to multi-robot system interfaces. Take for example switching on a robot or initiating a controller with a button push, while this works for one or two robots, there will come some point, perhaps when dealing with 50 or 100 robots, when this strategy will no longer be practical.

McLurkin et al. are familiar with the difficulties of maintaining and interacting with more than 100 robots [62]. In addition to presenting a data terminal interface, a more advanced graphical interface coined “SwarmCraft” is presented in their work: inspired by real-time strategy game interfaces from StarCraft and WarCraft (Blizzard, 1998), “SwarmCraft” allows remote control and debugging information of individually selected robots from the swarm. Fortunately their system contains navigation beacons which allows them to spatially organize their robots in a graphical tool. When extensive infrastructure such as ceiling mounted cameras exists, a live video feed can be used to present a real view of the world

including obstacles. The system presented by Kato et al. displays a live video feed on an interactive multi-touch computer table; users can control the robots' paths by drawing a vector field over top of the world [47].

On the other hand, systems with noisy localization, or worse without any localization can not conveniently be organized into a correct spatial representation of the world. It is conceivable to individually label each robot with a unique identifier (for example an IP address), which a user can then use to remotely connect to the particular robot. While this easy to implement technique works for experimental research and debugging, it becomes problematic when a user wants to quickly interact with *that* robot (for example the robot that keeps bumping into the user's feet) rather than robot #613. This concept is similar to deictic representation, as described by Agre and Chapman [1].

An earlier multi robot system presented by Payton et al. makes use of infrared (IR) LEDs for communication between robots among a swarm [78]. Each robot is equipped with eight directional IR transceivers that transmit orientation specific information for each transceiver; this elegant design effectively captures a vector flow towards some target pheromone without any map of the world. A remote control device also exists for communication with either the entire swarm via means of an omnidirectional IR LED[3] or a *particular* robot which is selected by the user by pointing a directional IR LED at the robot of interest. While this remote is not explicitly described in any of Payton's papers, Payton modestly pointed us (through personal communications) to slides from his presentation at the International Conference of Simulation of Adaptive Behavior (SAB) 2004 [77].

A similar system currently being developed by Naghsh et al. attempts to help firefighters navigate smoke filled buildings where visibility is limited or non-existent [70]. Much like Payton's work, an augmented helmet device is used to point firefighters to goals and more importantly exit points. The authors outline firefighter-specific human-robot swarm interaction issues; however, the problem of selecting a particular robot from the swarm is never addressed.

While swarms can scale to thousands of robots [98] humans can not effectively monitor or interact with such a large number of robots [73]. The work of Bashyal et al. questions how much of a swarm should a human interact with [4]. While they make a distinction between human-robot and human-swarm interfaces, their simulation-based experiments fail

[3]robots propagate broadcast messages to robots which are not in direct line of sight of the remote

to address the robot selection problem and simply make use of a point and click god-like interface for an unrealistic simulated environment.

A novel human-robot interface designed for cleaning robots is described by Zhao et al.[108]. Rather than interacting directly with the robots, they propose the user interacts with the environment by leaving "notes", such as "vacuum the floor" or "mop the floor" for the robots at work site locations. An overhead camera recognizes fiducials printed on the notes, and dispatches an appropriate robot to complete the task. A robot equipped with a printer is also used to leave notes for the user, e.g., when a failure occurs and the task can not be completed, the printer robot will leave a note with the appropriate error code.

2.5 Gesture based robot interaction

Gestures – intentionally choreographed body poses and actions – are commonly used in day to day human interactions [49]. Gestures are a form of non-verbal communication; they can be used to convey geometric and spatial information (e.g. pointing in a direction), as a visual representation of an object or action, or simply as a greeting (e.g. waving hello or good-bye). Gesture-based robot control systems have the advantage of allowing users to freely walk in the same environment as the robot, and can provide a natural interface for issuing commands.

Many techniques exist for recognizing gestures; however, computer vision methods have the advantage of being passive and do not require the user to wear any specialized hardware. There is a vast computer vision literature on the gesture recognition domain: Mitra and Acharya provide a survey [64].

Several gesture-based robot interfaces exist; we do not attempt to provide an exhaustive survey, but rather mention some interesting examples. Systems may use static gestures – where the user holds a certain pose or configuration – or dynamic gestures – where the user performs a combination of actions. The speed of dynamic, or motion-based gestures can be used to indicate the urgency of the command or task.

Static hand gestures are used by Becker et al. to command a robotic arm to pick up objects. The position of the hand is used to indicate which object to pick up, and the gesture of the hand, based on a certain finger configuration, specifies how to pick it up, e.g., from the top or side [7]. A similar gesture-controlled robotic arm is presented by Rogalla et al. in [85].

Both static and motion-based gestures are used in the work of Waldheer et al. They present a gesture based human-robot interface for directing a robot to pick up trash; the robot then autonomously drives to a garbage bin. They argue that static gestures are good for issuing commands such as "stop" which must be quickly issued and classified [101].

Work by Loper et al. demonstrates a mobile robot designed to follow a person in both indoor and outdoor environments. An active depth sensing camera is used to recognize static arm gestures, which are used to command the robot; the user can additionally use speech-based commands [56]. Earlier work by Kortenkamp et al. used an active vision system to track the position and configuration of a user's arm, which is then used to build a skeleton model. Static arm gestures are classified by comparing the angles of the skeleton joints to hard-coded values; in addition to recognizing an extended arm as a pointing gesture, a vector of the arm is extracted and can be used to direct the robot to a particular point [52].

A multimodal interface based on speech and gestures is presented by Perzanowski et al. in [80]. An active vision system is used to interpret pointing gestures as directional vectors, and to measure distance between the user's two hands. Even though their system allowed users to control the robot by joystick, or to specify an (x, y) coordinate on a PDA, they found that users preferred to direct the robot using natural language combined with gestures. In a subsequent paper [79], Perzanowski et al. consider a team of multiple robots, and discuss the possible use of gaze for directing an utterance at a particular robot; however, no evidence suggests they actually built such a system. The authors instead choose to give each robot a unique name which is used to direct an utterance at a particular robot, e.g., "Coyote, go to the north side of the nearest building".

All gesture-based systems discussed so far are designed to work with a single robot, with exception of the work of Perzanowski et al. described in the previous paragraph. Their system, however, made use of speech; there are no examples of gesture-based interfaces designed for multi-robot systems which rely solely on non-verbal communication. In this thesis, we present such a system: a user first selects an individual robot with face engagement, then uses motion-based gestures to command it. Our system allows a user to interact with multiple robots in a shared environment by only using visual cues.

Chapter 3

Robot selection

This chapter addresses the interaction between a human and a single robot located among a group of robots. Before assigning a task to an individual robot, the human operator must first somehow designate a particular robot of interest as the selected robot he or she will be addressing. We will refer to this as the *robot selection problem*: how does a user interact with a particular robot within a group of robots without accidentally selecting or issuing commands to multiple robots?

The difficulty of the robot selection problem depends on the particular human-robot interface of the system. For example, interfaces that have physical buttons or touch screens located on each robot are immune to the problem since there is no disambiguation when the user issues a command to the robot. In this case a private communication channel exists between the human and each robot since the user must physically approach and touch each robot. However, systems that do not have a private communication channel for each robot and rely on broadcasting commands through a shared medium are susceptible to the robot selection problem. These media include audio, infrared, radio and vision. Most systems assign a unique name or identifier which can be used to specify which robot the message is intended for. The Internet, for example, uses IP addresses to deliver a message over a shared medium to a particular host; however, while IP addresses are easy for computer-to-computer communication, long unique identifiers are not appropriate for HRI as suggested by Fig 3.1. Assigning names to each robot (similar to the way hostnames are mapped to IP addresses) would provide a more usable interface; however users would still have to learn each robot's name; furthermore, as the system grows to large numbers of robots, coming up with unique names may be an issue.

Figure 3.1: We suggest using face engagement is much more natural than a unique identifier

As mentioned in the previous chapter, eye gaze and eye contact are natural non-verbal communication channels used for showing attention. Our approach to the robot selection problem is focused on maintaining face to face communication between a human and an individual robot. Face engagement serves as the means for designating *that* robot (the one the user is looking at) as the selected robot. However, since face engagement occurs in a shared communication channel between the user and all robots within line of sight, the robots must collectively agree upon a single robot designation to ensure only one robot will ever respond to the user at any given time.

The human operator should be able to select and interact with a robot at different distances; however, implementing eye gaze on a mobile robot for use at larger distances can be a costly endeavour since the use of a telephoto lens or high resolution camera must be used to capture a high quality image of the human's eyes [102]. We on the other hand, use face detection rather than estimating eye gaze; this allows us to use smaller (and cheaper) cameras without zooming capabilities. Our system assumes only a single human will be interacting with the system at any given time; however, this single human will be simultaneously visible to multiple robots. Our system is designed to work at distances varying from 1 to 4 meters. The challenging aspect of our proposed solution to the robot selection problem is disambiguating which robot is currently being looked at through means of a distributed leader election algorithm based on the score of the detected face.

3.1 Face detection

The first phase of robot selection involves face detection. Each robot is equipped with a Lenovo ThinkPad R61 7744 laptop with an Intel Core 2 Duo 2.2GHz dual-core processor and 2GB of memory; we use the built in 640x480 resolution video camera to capture upward pointing images. Given an image such as the one presented in Fig 3.2, we are interested in locating a rectangular region in the image that contains a face. Furthermore, we want to extract a corresponding score indicating how likely is it that a frontal face has been detected.

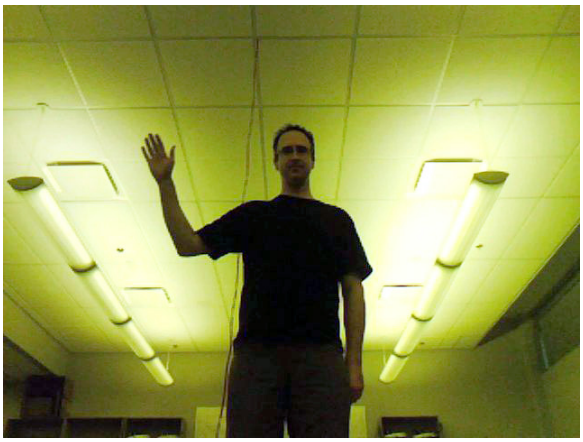


Figure 3.2: Sample video frame of a user looking at the robot

Faces are detected with the Viola-Jones method [99], a widely used[1] real-time object detection algorithm. We use an implementation provided by the OpenCV software library [10]. An overview of the algorithm, originally described by Viola and Jones is presented in the following section.

3.1.1 Viola-Jones face detection

The following sections provide an overview of the various components of the Viola-Jones method.

[1]Viola-Jones is the standard method for face detection in OpenCV, which reached 2 million downloads as of 2008 [10]

Haar-like features

Haar-like features, which are named after Haar wavelets [58] due to their similarity, are used to extract features from an image. Rather than working with individual pixels, Papageorgiou et al. suggested using Haar-like features as basis functions of the image [75]; however, real-time performance was never addressed in their paper. The Viola-Jones method makes use of Haar-like features; examples can be seen in Fig 3.3.

Each feature used by the Viola-Jones method is located relative to an enclosing detection window, shown as the outer border of the examples provided in Fig 3.3. The features' values are calculated by taking the difference of the summed pixel values of the white rectangle from the summed pixel values of the black rectangle.

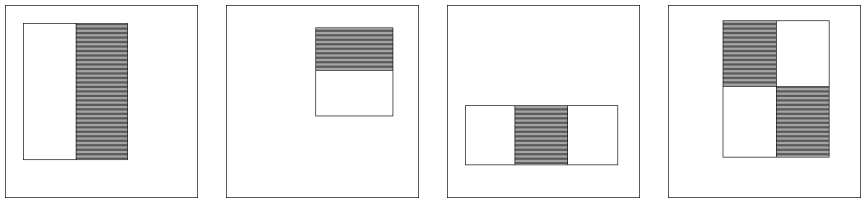


Figure 3.3: Example rectangle features used by the Viola-Jones method

Efficient Computation

One of the novel contributions of Viola and Jones is the efficient computation of Haar-like features. An *integral image* is first computed: each pixel (x, y) in the integral image represents the summation of all pixels within the rectangle formed between the two points (x, y) and $(0, 0)$ of the original image. Once this integral image has been calculated, each Haar-like feature can be computed in constant time by looking up the cached pixel sums at each corner and taking the appropriate differences[2]; this process is described in detail with excellent figures in their paper.

[2]rather than summing each individual pixel that lies in the rectangular area

(a) The detector is scaled and slid across the image

(b) Two Haar-like features used to detect faces. The one on the left matches a horizontal edge created by the eyes, and the one one the right matches vertical edges created by the nose

(c) Multiple positive sub-windows clustered around the face represent a high probability of a detected frontal face

Figure 3.4: Face detector

Sub-windows

These features are computed at many different scales and positions. During the offline training phase, an exhaustive set of features (reported size of 160,000) is calculated for each positive and negative 24x24 resolution training image. A classifier is learned with the adaBoost [27] machine learning technique. Once the offline learning phase is complete, the learned classifier can then classify an image by selecting a small subset of the exhaustive feature set; this reduces a significant amount of computation as only a smaller number of features must be calculated.

During the detection process a sub-window is effectively slid across the image at various scales and positions as depicted in Fig 3.4(a). Rather than resizing the image to 24x24 pixels which would be prohibitively expensive, the detector is scaled and repositioned to match the corresponding sub-window. This allows features to be computed in constant time with the integral image. The classifier accepts or rejects each sub-window as a detected face based on the calculated features. The example shown in Fig 3.4(b) shows the top two positively matched Haar features used to classify the image: the first feature corresponds to a horizontal edge formed by the eyes of the face, and the second feature corresponds to vertical edges formed by the nose.

A true face in an image will likely cause multiple overlapping sub-windows to return positive results; an example can be seen in Fig 3.4(c). This occurs because the detector is insensitive to small changes in scale or position of the sub-window. Recall that each feature

is calculated based on a summation of pixels; a small variation in the sub-window position will only effect the inclusion of pixels near the border of the feature rectangles; however, the majority of the pixels lie in the center of the rectangle and will equally contribute to the overlapping features' values.

These overlapping detected regions are most often clustered into a single detected rectangle. The number of overlapping regions can be used to assign a score to the detected face[3]. By default, the OpenCV object detector implementation rejects detected regions with fewer than three neighbours. We use the number of neighbouring regions to both reject low scoring regions and more importantly determine how frontal the detected face is.

3.1.2 Face score

The face detector is trained on frontal faces only; the training set[4] does not contain any images of face profiles. Therefore, the best matches occur when the detected face is looking directly at the camera. Using the number of overlapping neighbouring sub-windows as the score does not necessarily indicate how frontal the face is. An obscured frontal face, for example, may receive a lower score than a visible and well lit non-frontal face. However, if the *same* face is captured simultaneously by multiple cameras, then the scores *can* be used to detect the most frontal face. This observation is a novel contribution of this thesis.

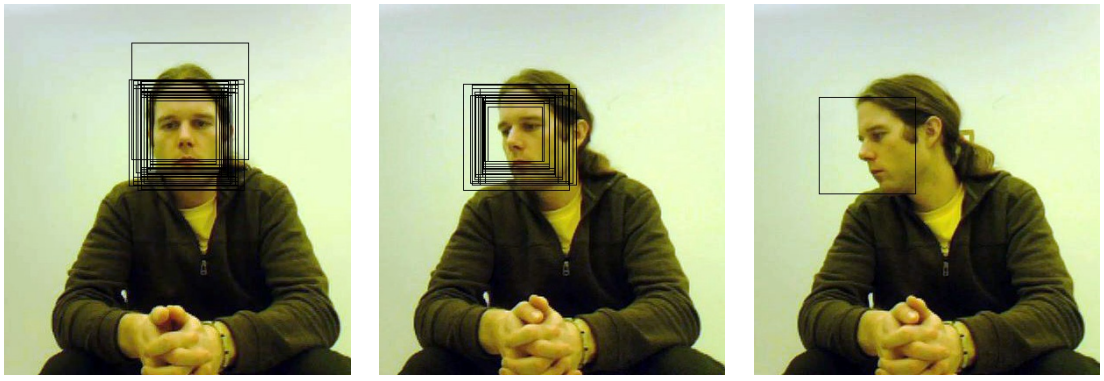
Fig 3.5 provides an example of three different images of a person looking in three different directions. A frontal face is captured in the first image (Fig 3.5(a)) which has the highest score; as the person looks away from the camera the score decreases. In the extreme case where only a profile of the face is captured, the face is barely detected and receives a very low score.

3.2 Leader election

The second phase of our solution to the robot selection problem is to perform a leader election algorithm; this ensures only a single robot will ever be designated as the selected robot. The election determines which robot is most likely being looked at "head-on" by the user, as estimated by the highest detected face score.

[3]The Viola-Jones classifier score could also be used; however, the OpenCV implementation makes it difficult to retrieve, and using multiple detections is arguably more robust.

[4]We use the pretrained frontal face Haar classifier supplied with OpenCV.



(a) front - score: 34 (b) 45 degree - score: 19 (c) side - score: 1

Figure 3.5: Candidate rectangles detected by the OpenCV Haar classifier cascade for frontal faces. The number of candidate rectangles are used to indicate how likely the face is a frontal face.

Since the user might be visible to multiple robots, it is crucial that only a single robot ever respond to the user at any given time. This in effect, requires some form of mutual exclusion among the robots, which are hereafter referred to as nodes. Our leader election algorithm must meet the following conditions, based on the conventional definition of mutual exclusion [94, p. 102]:

LE1: (safety) Only a single participating node can ever be elected at any given time.

LE2: (fairness) Each node shall have the opportunity to participate in an election, regardless of whether or not any node is elected.

LE3: (liveness) If a node exists with a score greater than any other node, it will eventually be elected.

The condition LE1 is motivated by our goal of individually controlling a particular robot in a group of robots; it is therefore required that no two robots ever simultaneously respond to a command which was addressed to a single robot. When a robot is first selected by a user with face-engagement, that robot begins to glow to offer visual feedback to let the user know it is ready to accept a command; the visual feedback must be mutually exclusive, i.e., only a single robot can ever glow at any given time.

Any robot may be selected by the user; therefore, the fairness condition, LE2, requires that each node is given a chance to run as a candidate in any election at any time regardless if any other node is currently elected. Each node must be capable of raising a "vote of no confidence" if it believes it should be elected. For example, as the user shifts his or her attention from the currently selected robot to another robot, the newly addressed robot must be able to force a new election in order to become the newly elected robot.

The last condition, LE3, states that eventually some robot must be elected – given no ties occur. While the nodes in our network are totally ordered, and therefore ties *could* be broken by comparing the unique IP address of each node, we choose *not* to break ties. Consider two robots placed side by side; if the user looks at the location between the two robots, both robots will likely have the same detected-face scores. Rather than have the network ordering arbitrarily break the tie, we choose not to elect any robot, and instead force the user to break the tie by repositioning him or herself – this gives the user the ability to select the robot they intended to select, rather than arbitrarily selecting a robot that the user may not have intended to select.

Given a robot does have the highest score, condition LE3 states that our system can not result in deadlock.

To solve this problem, we use a variation of the ring-based election algorithm first described by Chang and Roberts [16]. In their algorithm, a ring network with reliable communication is used to pass an election message in a circular manner around the ring. Each node is assumed to have a unique identifier (UID); in our case, we have a unique IP address. The general concept behind this algorithm is each node creates a message containing its UID and sends it to its neighbour. When a message is received at a node it compares the number to its UID and either:

1. passes the message to the next node if the number is greater than its UID,
2. discards the message if the number is less than its UID, or
3. declares itself the elected node if the number equals its UID.

A node will only ever declares itself elected after hearing its own message; this implies that its own message must have travelled completely around the ring, thus passing by *every* other node in the network. Condition LE1 is therefore met, since the node with the greatest UID will discard every other nodes message (because it contains a lower UID) before it has

the chance to return to its owner. The *elected* node's message will not be discarded by any other nodes since its number is greater than every other node in the network.

In the Chang-Roberts leader election, once a node has received its own message it declares itself the elected node and sends out an *elected* message around the ring indicating it has won the election (and therefore the election is over). Our variation of the leader election algorithm does *not* send an *elected* message and is described in the next section.

3.2.1 A continuous leader election

Our variation of the ring based leader election algorithm contains the following differences:

1. the election is only initially called by a predefined node,
2. the detected frontal face score is used in addition to the UID,
3. only the elected node is informed it has been elected; other nodes need not know which node was elected, but simply that they were *not* elected, and
4. a new election is called immediately by the elected node, which stays elected until it receives a *vote of no confidence*.

The initial election is called by a predefined node in the ring[5]. Once the *election* message has been created, no other nodes can create a new message. A node can however, modify the contents of the *election* message before sending it to the next node. In some sense, the *election* message is treated as a token: a node can only transmit a message when it has ownership of the token. And specific to this case, since the token *is* the *election* message, transmitting a message causes a token transfer.

When a node receives an *election* message, the number contained in the message is compared to its own number in a similar fashion; however, in this case the number used is the detected *face score*. Since the score is not guaranteed to be unique, the message contains both the detected face score and a UID, which are recorded as an *election tuple* (uid, s), where uid is a unique identifier, and s is the detected face score. When a node receives an *election tuple* it either:

[5]initiated by the ring owner; section 3.2.2 explains the ring creation process

1. recognizes itself as the elected node if the received tuple contains its own UID, and starts a new election,

2. passes the unmodified tuple to the next node if the contained score is greater than its own, or

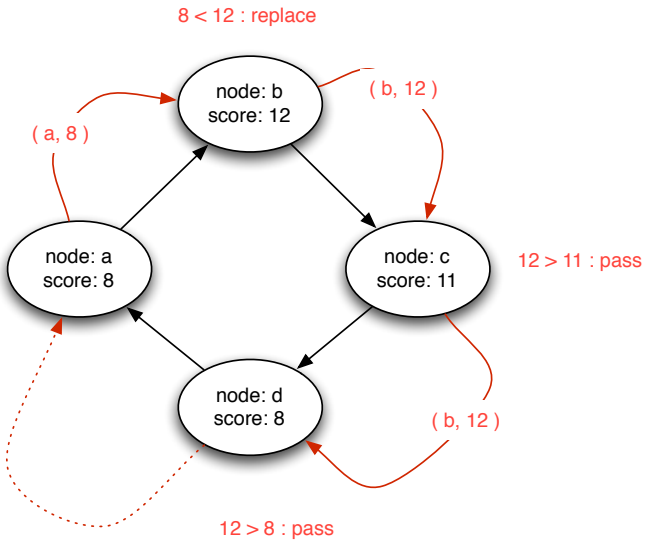
3. replaces the contents of the tuple with its own score and UID if the contained score is less than or equal to its own.

This message passing routine is described in Algorithm 1. When an *election tuple* is received at a node the `RECEIVE`$((uid_{rx}, s_{rx}), uid_i, s_i)$ routine is called, where (uid_{rx}, s_{rx}) is the received tuple, and uid_i and s_i are respectively the unique id and current score of the receiving node i.

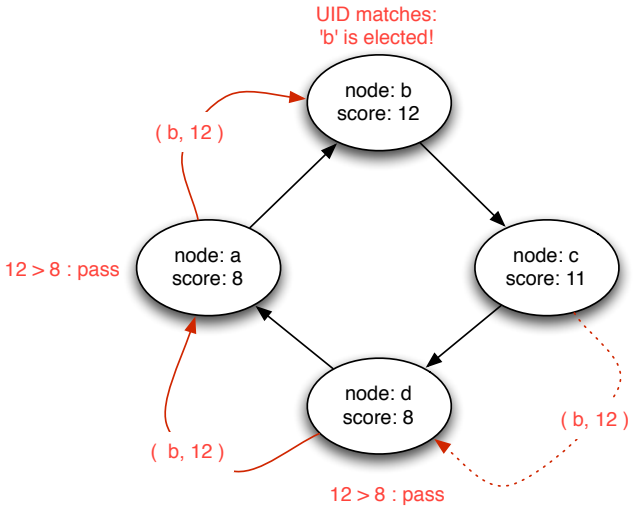
Algorithm 1 message receiving algorithm

```
1:  procedure RECEIVE((uid_rx, s_rx), uid_i, s_i)
2:      if uid_rx = uid_i then
3:          elected_i ← true                          ▷ the tuple completed a cycle
4:          SEND(uid_i, s_i)                          ▷ start a new election
5:      else
6:          elected_i ← false
7:          if s_rx > s_i then
8:              SEND(uid_rx, s_rx)                    ▷ pass unmodified tuple
9:          else
10:             SEND(uid_i, s_i)                      ▷ set node as highest score
11:         end if
12:     end if
13: end procedure
```

The example in Fig 3.6 shows the initial election being called by node a. Node a records its score of 8 and UID of a in the tuple $(a, 8)$ and passes it to node b, which has a larger score. Node b replaces the tuple with $(b, 12)$ to indicate it has a higher score, and forwards it to node c. Node c has a lower score and therefore forwards the tuple without any modifications. The tuple is passed from node to node and eventually ends up back at node b as illustrated in Fig 3.6(b). Since no other node in the ring has a greater score, the unmodified tuple $(b, 12)$ has made a complete cycle, thus visiting each node before returning to b. Node b recognizes itself as the node with the highest score and determines it is the winner.



(a) node a initiates the election



(b) the election continues until node b receives its own UID

Figure 3.6: The tuple (uid, score) is passed between nodes in the ring. A node compares the received face score with its own score, if the received score is larger than its own, it passes the unmodified message. If the received score is less than or equal to its own score, it replaces modifies the tuple with its own score and UID before forwarding the message. A node is declared the winner once it receives a tuple containing its own UID. We intentionally modify the message if the scores are equal to avoid electing any winner in the event of a tie.

Condition LE1 (safety) is still met with our modifications; at most only a single node will ever be elected the winner. Before a node can receive its own message containing its UID, it must pass by every other node in the system. In order for the UID to be preserved, every other node must have a smaller score – otherwise the tuple would have been modified before completing the cycle, resulting in a UID mismatch.

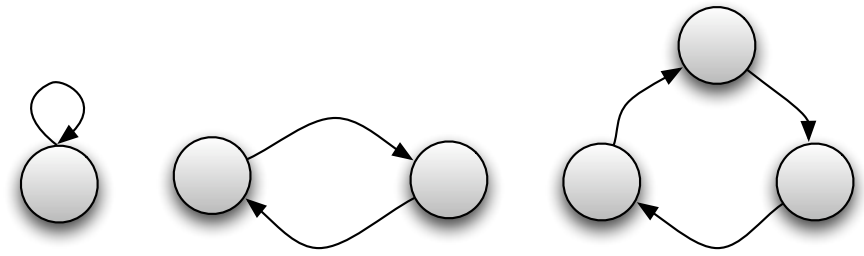
The liveness condition, LE3, is met since each node must pass on the message – either by replacing the tuple with its own score and UID, or by passing the unmodified tuple. If two or more nodes have equal scores, each node will replace the UID of each other's modified tuple; this prevents any UID from completing a cycle, thus never allowing any node to be elected during a tie. This will *not* result in a deadlock since the scores are *dynamic*, and will change as the user repositions his or her face. This strategy ensures small variations in the face score does not cause robot selection to sporadically jump between two robots.

In order to meet condition LE2, our system must be fair to all nodes and give each node the opportunity to partake in the election process – regardless if any other node is currently elected or not. If a non-elected node receives a higher score than the currently elected robot, e.g., the user looks away from the elected robot to focus on a different robot, then that node must be able to become the newly elected node. To achieve this, once a node is elected it immediately calls a new election. It creates a new tuple with its UID and its most up-to-date face score. If this tuple reaches a node that has, in fact, a *higher* score, then, like before, the *election tuple* is updated with the higher score and corresponding UID. Once the modified tuple reaches the currently elected node (which started this new election) it notices a different UID contained in the tuple and considers this as a *vote of no confidence*. Upon receiving the *vote of no confidence*, the node marks itself as un-elected and passes the unmodified tuple to its neighbour; the tuple is passed in the same manner until it completes a cycle, resulting in a newly elected node.

3.2.2 Ring network

In this section, we describe how the ring network is constructed. Our implementation focuses on creating a unidirectional ring network: adjacent nodes are connected in a circular arrangement to form a closed loop. Each node is connected to two neighbours: one which it receives from, and the other it sends to. Sample ring networks can be seen in Fig 3.7.

Token passing in a ring network can be used to provide mutual exclusion between nodes on a ring. A node may only enter its critical section, i.e., a section of code which must



(a) 1 node (b) 2 nodes (c) 3 nodes

Figure 3.7: Examples of different sized unidirectional ring networks

be executed separately from any other node, when it has ownership of the token. Once the critical section has been executed, the token is released and passed to the next node in the ring. Our system is similar to a token-ring network [14], in that only a single *election* message is ever transmitted on the ring at a given time; nodes can only announce a higher face score when they have ownership of the election message.

Our ring network is implemented on top of an Ethernet-based TCP/IP network; nodes communicate using TCP and UDP and are located on the same subnet. The goals of our ring network are:

RN1 no a priori information is required by any of the nodes, i.e., nodes are assigned dynamic IP addresses and are not given any address to initially connect to,

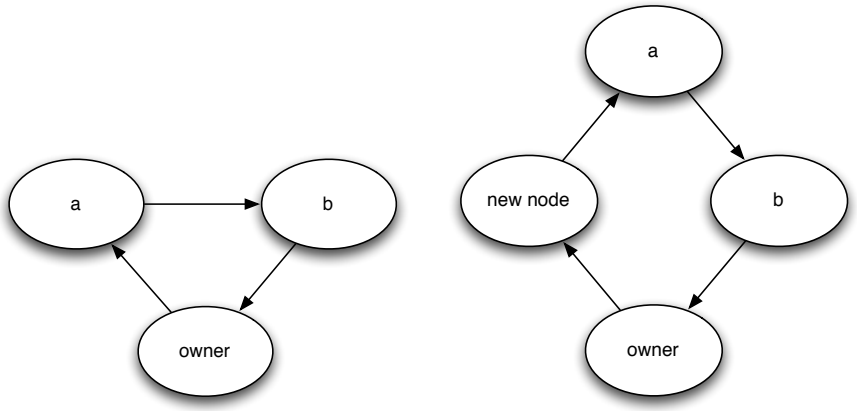
RN2 the ring has a unique *owner* node which is responsible for calling the initial election, and

RN3 multiple rings should eventually reform into a single ring.

Ring Owner

Each ring has a *owner* node which is the coordinator of the ring. When a new node wishes to join an existing node, a join request is sent to the ring owner. The owner responds to the node's request with the IP address of a neighbouring node to forward messages to, and a unique token identifier used to verify received messages originated from the correct ring [6].

[6] as supposed to an incorrectly sent message from a previously crashed ring



(a) before (b) after

Figure 3.8: Joining a ring

The node insertion procedure is illustrated in Fig 3.8. A new node is always inserted directly in front of the ring owner – the new node will receive messages directly from the ring owner. The newly inserted node will then forward messages to the whichever node the ring owner previously sent messages to.

A ring is guaranteed to have a single owner. During node initialization, nodes can do one of two things: 1) form a new ring, or 2) join a preexisting ring. Nodes which form a new ring become the ring owner. The ring has an initial size of 1; messages are sent directly from the owner to the owner as shown in fig 3.7(a). As nodes join a preexisting ring, they are assigned positions in the ring, but ring ownership is non-transferable. This ring owner can be responsible for calling the initial election upon ring creation, and thus our goal RN2 is met.

Ring Creation and Membership

When a node is first initialized, it has no knowledge of other nodes, i.e., no addresses of existing rings are known. Each node is assigned a dynamic IP address (via DHCP). We assume each node is located on the *same* subnet, and a common port is known to all nodes. Nodes can therefore communicate via broadcast UDP messages using the predefined port.

Once initialized, the newly created node passively listens for broadcast messages for a set number of seconds. The owner of each ring is responsible for periodically broadcasting a

message advertising the existence of its ring. In our implementation, the broadcast occurs every 2 seconds; therefore, the newly created node must wait double that time, 4 seconds, to ensure a message is transmitted at least once. However, since this message is broadcast over UDP, there are no guarantees the message will be heard during that 4 second window.

One of two possible outcomes occur: 1) one or more messages are received that advertise the existence of a ring, or 2) no messages are received. The first approach which comes to mind is to join the first detected ring; however, we take a different approach: if a ring is detected which is owned by a *lower* IP address than its own, it will request to join the ring with the *lowest* IP address. If, however, no messages were heard from lower IP addresses (or none at all), then the node creates a *new* ring which it begins to advertise.

Once a node has joined, or created a ring, it continues to monitor broadcast messages for the existence of other rings. If at any point, it detects a ring advertised by a *lower* IP address than its ring's owner, then it shall immediately disconnect – regardless if it is the ring owner or not – and join the newly detected ring with the *lowest* IP address.

By broadcasting the existence of rings over UDP message, we are able to reach goal RN1, with out any a priori knowledge except for a predefined port number. Furthermore, by favouring membership in rings which are owned by the lowest IP address, multiple rings will eventually reform into a single ring, thus meeting our goal RN3.

An overview of creating or joining a ring is given in Algorithm 2. Upon initialization, the node either calls `CreateRing` or causes `Join` to be called on the ring owner's node.

Implementation Details

As mentioned previously, UDP messages are broadcast to advertise the existence of rings; however, these broadcast messages are *not* guaranteed to be reliable. However, in practice, we did not encounter any large loss of broadcast messages during our experiments. Nonetheless, once a node has detected a ring, all subsequent connections are performed over a reliable TCP connection.

Election messages are encoded using JavaScript Object Notation (JSON) [19], which is a text-based data exchange format. A sample election message is shown in table 3.1. The election message is then sent to the next node in the ring via a JSON-based remote procedure call (RPC), as specified in the JSON-RPC specifications[34]. An example of the data sent is shown in table 3.2.

Algorithm 2 Ring membership and creation algorithms

```
1:  ring_owner ← ∞                                   ▷ the UID of the ring owner
2:  is_owner ← false                                 ▷ Boolean indicating if node owns ring
3:  next ← NULL                                      ▷ which node to forward to
4:  my_uid be the node's address
5:
6:  procedure ReceiveBroadcast(i)    ▷ called upon hearing node i advertising a ring
7:      if i < ring_owner then
8:          disconnect from current ring
9:          request to join ring i
10:     end if
11: end procedure
12:
13: procedure Join(i)                ▷ called by other nodes wanting to join this ring
14:     if is_owner = false then
15:         return an error
16:     end if
17:     next_i ← next
18:     next ← i                     ▷ ring owner now forwards messages to the joining node i
19:     return next_i                ▷ return address of node the joining node will forward to
20: end procedure
21:
22: procedure CreateRing                             ▷ create a new ring
23:     next ← my_uid
24:     is_owner ← true
25:     ring_owner ← my_uid
26: end procedure
```

Table 3.1: Sample election message encoded in JSON. The UID is referred to as "owner"; "score" represents the detected face score

```
{"owner":"192.168.1.101","score":47}
```

Table 3.2: Election messages are sent between nodes using JSON-RPC

```
{ "id":15432, "method":"token", "params":{ "tokenid":"192.168.1.100",
        "value":{ "owner":"192.168.1.101", "score":47} } }
```

3.3 Conclusion

In this chapter we described a computer vision-based method to solve the robot selection problem. We used a well-known face detector trained to detect frontal faces, and observed that the score of the detected face can be used to estimate which camera the user is looking at. The score is used in a distributed leader election algorithm to elect at most a single robot, which will then respond to the user. Its use is demonstrated in chapter 6.

Chapter 4

Motion-based gesture recognition

In this chapter we describe a real-time system for gesture recognition; these gestures are interpreted as commands for the selected robot. The work discussed in this chapter is based on the paper "Real-time Motion-based Gesture Recognition using the GPU" by Bayazit, Couture-Beil, and Mori [6], which was presented at the IAPR Conference on Machine Vision Applications in Yokohama, Japan.

Given a live input video-feed, we derive a set of motion features based on smoothed optical flow estimates. A user-centric representation of these features is obtained using face detection, and a classifier is learned to discriminate between gestures. We develop a real-time system using GPU programming for implementing the classifier. We provide experimental results demonstrating the speed and efficacy of our system. Finally, we describe how this general gesture recognition system was retrained to be used on a mobile robot.

4.1 Introduction

Human gesture recognition in image sequences has many applications including HRI, HCI, and surveillance. We describe a system for real-time gesture recognition that uses motion cues to discriminate between different gestures. Examples of the set of gestures we use in our experiments are shown in Fig. 4.1. Consider the video frames shown in Fig. 4.2, these show examples of motion cues that are used to discern which gesture is being performed.

In this work we focus on recognizing gestures – intentional, choreographed motions performed by a co-operative subject. The main contribution of this work is developing a real-time system for gesture recognition based on optical flow features. The core algorithm

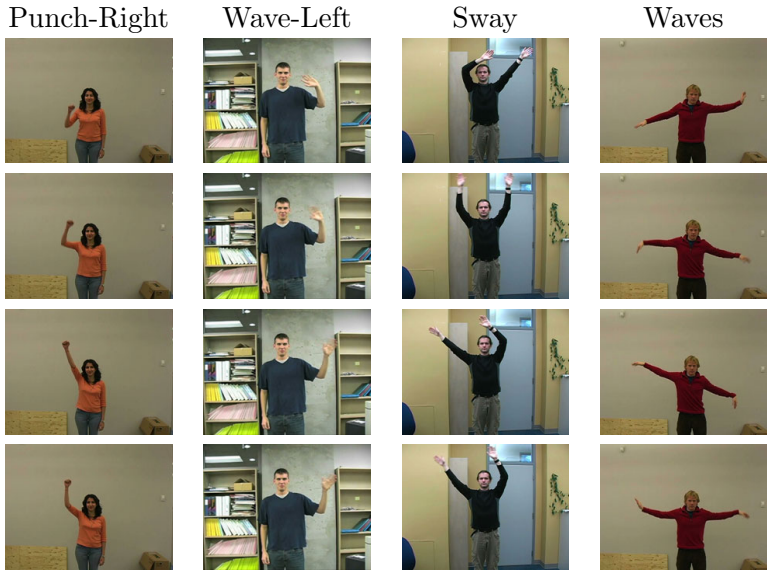
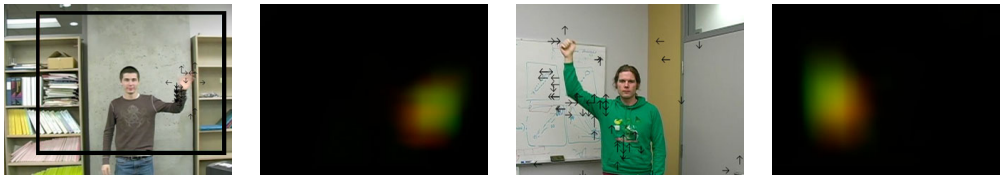


Figure 4.1: Example frames from a subset of the gestures used in our experiments.

for recognition is based on the work of Fathi and Mori [25], in which discriminative portions of optical flow are learned for recognizing actions. We use a variant of this algorithm, and combine it with a standard face detector [99] to localize the human figure in a video. We use the CUDA GPU programming API to create an efficient implementation of our action recognition algorithm. This results in a real-time system that can be used for applications in human-computer interaction. We demonstrate experimentally that it is fast and effective for gesture recognition.

There is a vast computer vision literature in the "looking at people" domain. Moeslund et al. [65] and Mitra and Acharya [64] provide surveys of this area, in general and as it pertains to gesture respectively. Closely related pieces of work in the motion-analysis vein include Bobick and Davis [8], who represent global silhouette shape and motion using *temporal templates*. Shechtman and Irani [88] develop a motion-consistency method that avoids aperture effects. Efros et al. [22] recognize the actions of small scale figures using features derived from blurred optical flow estimates. In contrast to these methods, which are based on nearest-neighbour classification, we learn an efficient classifier suitable for real-time applications. Further, our classifier contains the specific parts of motion that are important for discrimination between gestures, and previous work has demonstrated their success on



(a) (b) (c) (d)

Figure 4.2: (a,c) Input frames overlaid with an illustration of motion features used to discriminate this gesture from others. The heaviest weighted motion features chosen by our algorithm are shown as arrows noting direction of motion. A face detector identifies a rectangular region (a) which is used to crop a user-centric image (c) before calculating the corresponding optical flow (b,d). Colour is used to denote direction of motion.

standard action recognition datasets [25].

Another group of methods analyzes the motion trajectories of skin-coloured blobs (typically hands, sometimes with face) [83, 103]. Reliably detecting and tracking hands, especially with fast motions, can be challenging. In addition, our optical flow-based method can incorporate motion cues beyond those in just the tracked hand regions.

Real-time systems include Ike et al. [38], who develop an efficient hand gesture system using multi-core processors. Our approach shares similarities, but is based on motion rather than shape and uses the CUDA API for efficient computation on the GPU.

4.2 Algorithm

The algorithm presented here is based on the work of Fathi and Mori [25]. In that work, which can handle data acquired from a moving camera, a stabilized human figure-centric representation is obtained by running a pedestrian detection algorithm. Human actions are then recognized using a classifier learned from optical flow features.

In our algorithm, motion features are first computed on the input image sequence (stationary camera assumed). A standard face detector is then employed to obtain a user-centric representation, and again a classifier to discriminate between gestures is learned using a variant of AdaBoost. A real-time version of this classifier is deployed using the GPU. In the following sections we provide the details of this algorithm.

4.2.1 Motion features

To calculate the motion features, we first compute the optical flow for each frame. The optical flow vector field F is then split into horizontal and vertical components of the flow, F_x and F_y, each of which is then half-wave rectified into four non-negative channels F_{x^+}, F_{x^-}, F_{y^+}, F_{y^-}, similar to the method of Efros et al. [22]. We add another channel corresponding to motion magnitude F_0 which is obtained by computing the L_2 norm of the four basic channels. These five non-negative channels are then normalized to facilitate gesture recognition in soft-real time where frame rates can be variable, and to account for different speed of motion by different users. Given a vector v that represents the optical flow for a given pixel, compute $v' = \frac{v}{||v||+\epsilon}$, where ϵ is used to squash optical flow vectors with very small magnitude, most likely introduced by noise. In experiments we set $\epsilon = 0.5$.

Next, each of the five channels is box-filtered to reduce sensitivity to small translations by the user performing the gesture. This final set of five channels: $\hat{F}_{x^+}$, $\hat{F}_{x^-}$, $\hat{F}_{y^+}$, $\hat{F}_{y^-}$, $\hat{F}_0$ will be used as our motion features for each frame.

We represent a gesture as a collection of movements required to complete a single period of the gesture, rather than just capture a subset of the gesture phase. Hence, we aggregate the motion features over a temporal history of the last k frames, for some k which is large enough to capture all frames from a gesture phase. In practice, we set k to be the frame rate of our capture data; in other words, we assumed a gesture's phase was less than 1 second. Setting k too high will increase the classification response time when switching from one gesture to another. Setting k too low would increase the sensitivity of our algorithm to noise and variations in users' gesture speeds.

4.2.2 Face detection

Since we are working with an input video where the location of the user is unknown, to use the above motion features for gesture recognition we must obtain a coordinate frame relative to the user. Face detection is used to create a normalized, user-centric view of the user. The image is scaled based on the radius of the detected face, and is then cropped and centered based on the position of the face.

The frame is converted to grayscale and can be resized to a smaller resolution to speed computation time.

In our experiments, we crop and resize to a 30×40 pixel region centered around the user.

All five motion feature channels described above are flattened into a single vector $v \in \Re^{6000}$ ($6000 = 30 \times 40 \times 5$), which will be used to determine the gesture being performed.

4.2.3 Classification

The aforementioned motion features describe the user's entire motion. While these could be used directly for classification (e.g. nearest neighbour [22]), recent work [25] has suggested that learning a discriminative classifier from these features can highlight the important motions for characterizing particular gestures. Further, learning a classifier that uses a subset of the motion features, and does not have to compare to all training data, will lead to a more efficient system.

Given the labelled training data, we have a multi-class classification problem, separating the data into the different provided gesture classes. In this work we use the multi-class boosting algorithm AdaBoost.MH [87][1] that takes the motion features v as input. The supervised training is based on a set of labelled gestures. In the usual fashion we define a set of weak learners that are based on thresholding a value from a particular component of the motion feature vector v. For a gesture class l, each weak learner $h_t(v, l)$ is of the form:

$$h_t(v,l) = \begin{cases} 1 & \text{if} \quad p_{t,l} v_{\tau(t)} > p_{t,l}\theta_t \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

for a motion feature v, where $\tau(t)$ selects a component of the feature vector, $\theta_t \in (-\infty, \infty)$ is the classification threshold of the classifier, and $p_{t,l} \in \{-1, +1\}$ is a parity for the inequality sign.

The output of the final strong learner on motion feature v for class label l is:

$$H_t(v,l) = \sum_{t=1}^{N} \alpha_t h_t(v,l) \tag{4.2}$$

where α_t are the weights chosen by AdaBoost.MH. The maximum output value can be found, or, as in our experiments, these values can be used to produce precision-recall curves[2] to analyze the sensitivity of our algorithm.

[1] We used the implementation available at `http://multiboost.sourceforge.net/`.

[2] For an overview of precision-recall curves, and other evaluation techniques, see chapter 8 of [59].

4.3 Implementation

Our implementation uses the OpenCV library [10], which provides an implementation of Viola-Jones [99] face detection, and Horn and Schunck optical flow [36]. We assume a multi-core system and dedicate a thread for face-detection. This non-real-time thread runs at a lower FPS than the main thread; however, since our gestures have minimal torso movement, it is acceptable to create a figure-centric frame based on the last known face position from some previous frame.

4.3.1 CUDA classification optimization

The classifier, which accounts for roughly 10% of CPU time, computes the summation of a set of independent weak-classifiers. Blocks of 512 weak classifiers for a given class l are computed in parallel with the CUDA GPU programming API [71].[3]

The four parameters associated with each weak classifier $h_t(v, l)$ are stored as elements of the arrays:

```
float alpha[ N ], theta[ N ]
int   parity[ N ][ K ], column[ N ]
```

where the $column[t]$ stores the value of $\tau(t)$ used to select the feature vector component, N is the number of weak classifiers, and K is the number of classes. The parity values are encoded as boolean values by mapping $\{-1, 1\}$ to $\{0, 1\}$ respectively. Our implementation used $N = 1024$, and $K = 7$.

The classification is computed for a given 512 block and class l with the work kernel:

[3] A similar approach can use SIMD instructions on the CPU.

```
__global__ void classify_kernel(
  float* votes, int l, int offset )
{
  extern  __shared__  float d[];
  const unsigned int tid = threadIdx.x;
  const unsigned int t = tid + offset;

  d[ tid ] = data[ column[ t ] ];
  d[ tid ] = d[ tid ] > threshold[ t ];
  d[ tid ] = d[ tid ] == parity[ t ][ l ];
  votes[ tid ] = d[ tid ] * alpha[ t ];
}
```

where *votes* is an array of length 512 used to save the votes, and `offset` is a multiple of 512 used to select the appropriate block to compute. Upon completion, the individual votes stored in `votes` are summed via the reduce algorithm provided as a sample in the CUDA SDK. This process is repeated for each class parity $p \in \{0, 1, ..., K-1\}$ and block offset.

Fig. 4.3 compares the GPU and single-threaded CPU implementations on an Nvidia 9800GX2 and a dual-core Intel Xeon 3.2GHz. Each classifier was sampled 30 times, and ranges from 512 to 8192 weak classifiers. The CUDA-based implementation provides a speed-up factor of roughly four compared to the CPU method.

4.4 Results

We created a dataset consisting of seven gestures (punch-left, punch-right, sway, wave-left, wave-right, waves, and idle) performed by ten different people. The videos were recorded indoors against various backgrounds at 29.97 frames per second (FPS) at 720×480 with a stationary Canon GL2 camera. Each gesture lasted from five to ten seconds, and included a minimum of five continuous cycles of the action. Samples of our dataset are displayed in Fig. 4.1.

The waves and sway gestures produce motion in the lower and upper sections of the frame respectively, and similar motion around the torso. Two pairs of gestures: punch-left and wave-left; and punch-right and wave-right produce similar motions on their respective

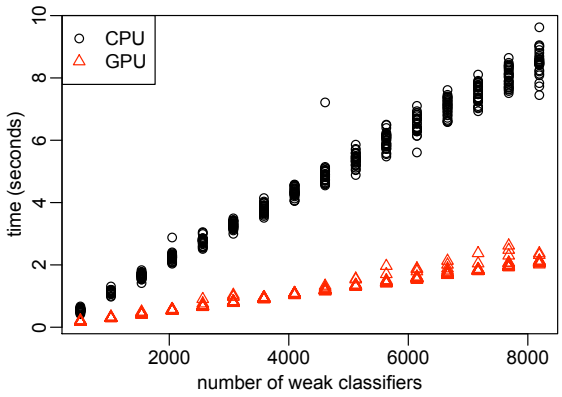


Figure 4.3: A parallel classifier implementation using the CUDA GPU programming API provides a speed-up factor of four compared to the single-threaded CPU implementation.

Table 4.1: Calculating optical flow on a smaller resolution significantly improves efficiency, measured in frames per second (FPS), while not significantly affecting classification accuracy.

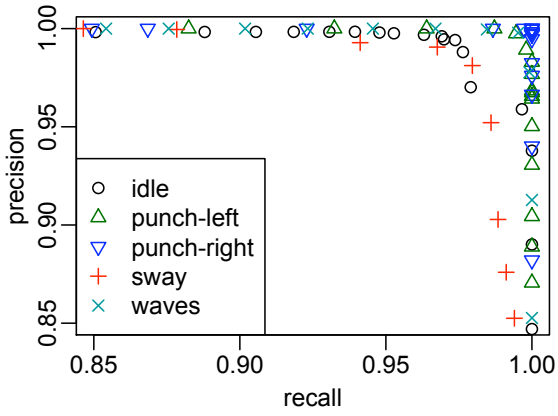
resolution	accuracy	FPS mean	FPS std dev
160×120	0.867	38.0	1.127
320×240	0.873	20.7	0.831

sides of the frame. While it is easy to distinguish a left pair from a right pair, determining a wave from a punch may be more challenging.

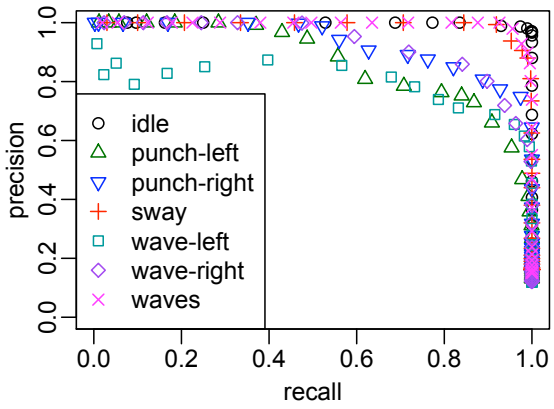
We achieved real-time results by resizing the user-centric frame before optical flow computation. We randomly selected a set of 30 videos from our dataset, and measured the FPS based on the processing time for the complete video. Table 4.1 show a smaller resolution gains a significant speed-up with minimal loss of accuracy, where accuracy is defined as the percentage of the time the correct gesture receives the highest classification. The remainder of this section is dedicated to a more detailed analysis of our results, which were obtained using a resolution of 320×240.

We tested the classifier by performing leave-one-out cross-validation on three different configurations of our dataset.

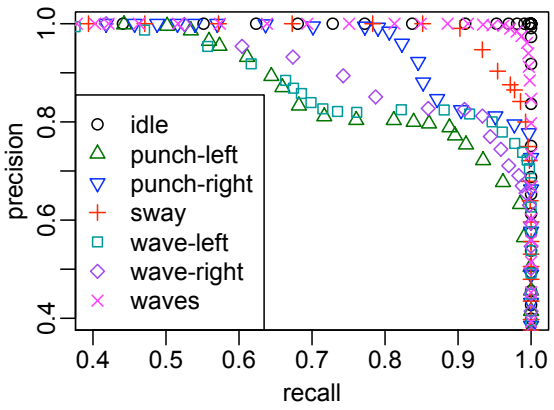
1. The first configuration was a subset of gestures that eliminated the wave-left and wave-right gestures. The punching gestures can easily be identified by motion on either the



(a)



(b)



(c)

Figure 4.4: Precision-recall graphs for different configurations of the gesture dataset. (a) a set of five gestures from distinct locations of the frame is performed by all ten subjects, (b) all seven gestures are performed by all ten subjects, (c) all seven gestures are performed by seven subjects who have minimal horizontal motion in the punching gestures.

Table 4.2: Confusion matrix for all seven gestures performed by all ten subjects. Rows represent the true action, and columns represent the percentage of actions returned by the classifier.

	sway	idle	waves	punch-left	wave-left	punch-right	wave-right
sway	1.00	0.00	0.00	0.00	0.00	0.00	0.00
idle	0.00	0.99	0.00	0.00	0.00	0.01	0.00
waves	0.06	0.00	0.94	0.00	0.00	0.00	0.00
punch-left	0.00	0.00	0.00	0.87	0.13	0.00	0.00
wave-left	0.00	0.00	0.00	0.32	0.68	0.00	0.00
punch-right	0.00	0.00	0.00	0.00	0.00	0.93	0.07
wave-right	0.10	0.00	0.00	0.00	0.00	0.14	0.76

left or right side of the frame; however, the sway and waves gestures produce motion in overlapping areas. The precision-recall graph in Fig. 4.4(a) verifies our algorithm is capable of recognizing this subset of gestures with both precision and recall over 95%.

2. The second configuration we tested contained all gestures, including the two waving and punching pairs. The classification of these pairs proved to be difficult, as shown in Fig. 4.4(b). The confusion matrix in table 4.2 shows we were able to distinguish a left pair from a right pair, and only confused the gestures within a given pair.

3. After reviewing our dataset, we noticed that three of our subjects punched horizontally to the sides rather than vertically. It is not surprising that those three subjects had the poorest results; they were trained against seven vertical punchers and only two horizontal punchers. Fig. 4.4(c) confirms that limiting our dataset to the seven vertical punchers, and thus employing a stricter control over the gestures, increases our results. The disadvantage of this; however, is that we must provide more detailed instructions to our subjects.

4.5 Retraining for use on a mobile robot

The dataset used for our experiments was recorded at the eye-level of the subjects; however, the robot-mounted camera is only 50cm from the ground. As a result, the input video from the robot captures the subject from a different perspective, as seen in Fig 4.5. Therefore, it is necessary to retrain the classifier with images captured from the robot's perspective.

(a) wave-left (b) wave-right (c) idle

Figure 4.5: Example frames from our robot-specific dataset used for training

Our robot-specific dataset consists of the gestures: wave-left, wave-right, sway, and idle. An additional negative "junk" gesture, which contains people walking around, is used as a negative-class to prevent false-positives. Each gesture is performed for 30 seconds by five different subjects.

4.6 Conclusion

In this chapter we presented a real-time gesture recognition system capable of classifying gestures. We employed a standard face detection algorithm to give a user-centric coordinate frame in which motion features were used to recognize gestures. Our system is capable of real-time performance. In particular, we use the CUDA API for GPU programming to obtain an efficient implementation of our classifier. One could also use the GPU for the face detection and optical flow computation, the other major processing needs of our system.

Our gesture recognition system is used by the robots to interpret gestures as commands; a demonstration is presented in chapter 6.

Chapter 5

The robot: putting it together

In this chapter, we describe how the physical and software components fit together to achieve a functioning mobile robot. In addition to building the robot, we describe a task, which is used to demonstrate our human-robot interface.

5.1 The physical robot

We use an iRobot Create as the base of the system, as seen in Fig 5.1. The Create is a controllable mobile robot platform targeted at educators, researchers, and hobbyists. The Create includes left and right bumpers on the front for sensing collisions, cliff sensors to prevent the robot from falling down stairs, a wheel-drop sensor for detecting if the robot has been picked up, and a single short IR sensor mounted on the right side for right wall following. The Create can *detect* collisions by using the front bumper; however, there are no sensors for *avoiding* collisions.[1]

One of the ongoing projects at the Autonomy Lab at Simon Fraser University is the Chatterbox project: the goal of the project is to build a swarm of fully autonomous robots which manage their energy resources and recharge when necessary to stay "alive". Jens Wawerla, a member of the lab, has been a major driving force behind the project. Jens developed a custom-made integration board for extending the sensors and computational power of the Creates. During the summer of 2009, members of the lab participated in a "build-fest" involving three consecutive exhilarating days of soldering and component

[1] except for the single sensor used for right-wall following

Figure 5.1: Out of the box iRobot Create

assembly; the process and results are shown in Fig 5.2. To date, 16 customized Creates, hereby referred to as Chatterboxes, can proudly be seen whizzing and whirring about in our lab.

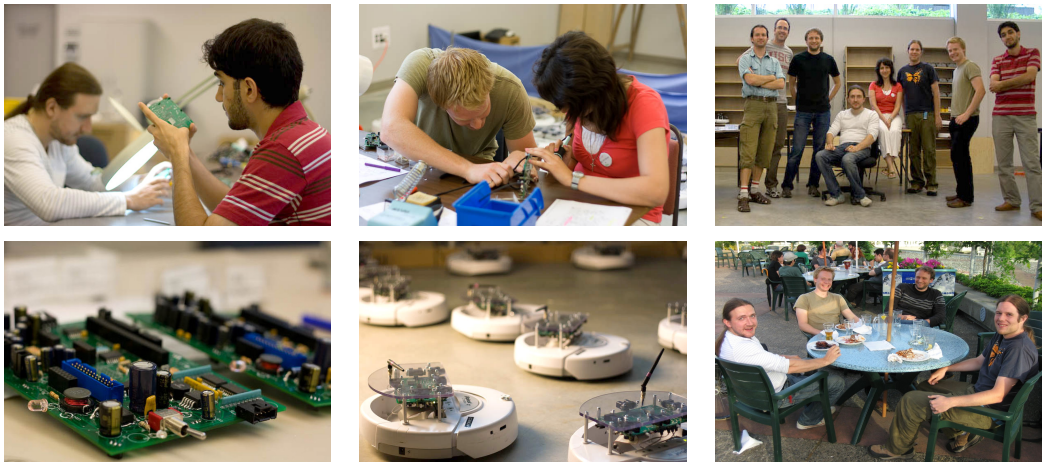


Figure 5.2: Photos from the Chatterbox build

The Chatterbox's integration board, seen in Fig 5.3 is controlled by a gumstix single board computer, measuring 2x8cm. Features include an 802.11 wireless network interface, five colourful LEDs, which we use for user-feedback, and six IR range sensors, used for obstacle avoidance. The gumstix computer runs Linux, and controls the Create through a serial connection.

One feature *not* provided by the Chatterbox is a video camera. Since the gumstix computer, which is designed for low power consumption and a small size, does not offer

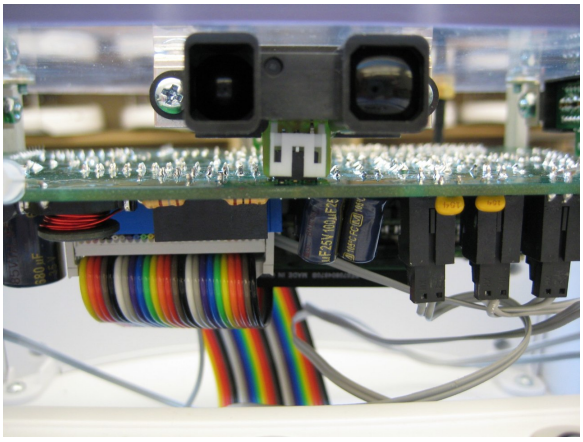


Figure 5.3: Gumstix integration board attached to an IR sensors

enough computational power for video processing, we have chosen to mount a laptop with a built in video camera on each of the three robots used in our demonstration. The final robot configuration is shown in Fig 5.4.

5.2 Demonstration task

To demonstrate our system, we perform a robot navigation task. Three robots and a human operator are located in a 7x10 meter room clear of static obstacles[2]. The robots:

1. first approach the user, who is located at a predefined location,
2. wait to be selected by the user,
3. receive a command, and
4. travel to a predefined zone which corresponds to the command they were issued.

Robots either travel to a *red* zone or a *green* zone which corresponds to the received gesture: *wave-left* or *wave-right* respectively. Upon reaching the two meter wide circular zone, each robot then returns to the user to await a further command. Red and green coloured lights are placed at each zone to represent the zone location for the users – these

[2]the existence of other robots and the user still poses a navigational issue

Figure 5.4: Create with custom sensor board and laptop

lights are *not* used by the robots in any way. The robots use a global coordinate system to locate these zones; three unique fiducial markers are placed on the walls near each target zone to aid robot localization, which is described in section 5.3.1. An overview of the zones and room layout is shown in Fig 5.5.

5.3 The robot controller

The robot controller runs under Linux on the gumstix computer and interfaces with the Create through the Autolab RAPI[3] library. The primary task of the controller is to perform navigation, but also includes gluing together the different components of the system; all visually sensed information is first captured and processed by the laptop and then sent to the controller over the wireless network. The laptop operates in one of two modes:

1. robot-selection and gesture recognition mode, as described in chapters 3 and 4, which is activated when the robot is stationary and awaiting a task from the user, or

2. fiducial detection mode, which is activated when the robot is navigating to or from any of the zones, i.e., performing work.

[3]Robot Application Program Interface

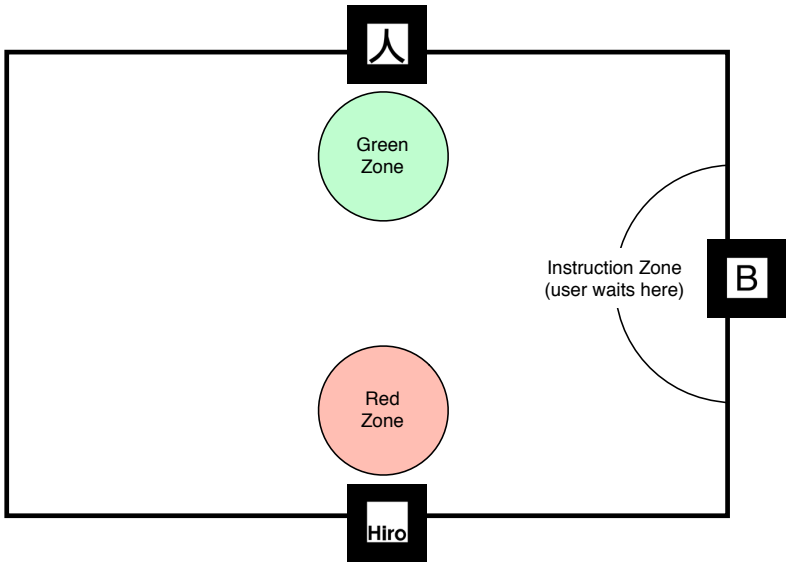


Figure 5.5: Robots first begin in the larger instruction zone where they are selected and assigned tasks by a user. Then, depending on the assigned task, robots either travel to the red zone, or the green zone. Fiducial markers are placed on the walls near each zone.

The laptop does not keep track of the robot's state, but rather acts as a sophisticated sensor. It can sense face engagement and classify motion-based gestures, *or* it can be used to sense fiducial markers; however, it lacks the processing power to perform both simultaneously. Therefore, the controller is responsible for switching the laptop, via a remote procedure call, into the appropriate state as needed. A break-down of the responsibilities of each device is shown in table 5.1.

Table 5.1: Responsibilities of the controller and laptop

Controller		Laptop
obstacle avoidance		leader election
navigation	← RPC →	gesture recognition
driving LEDs		fiducial tracking

Communication between the robot and laptop occurs over the wireless network and uses JSON-RPC. Each robot is assigned a static IP, which is known by the corresponding laptop

– the reverse is not true. Once the computer vision program starts, the laptop connects to the robot and identifies itself as the client. The controller can then set the laptop into either mode: *gesture* or *fiducial*.

Fig 5.6 provides an example of a typical communication exchange. We assume the robot is already located in the "instruction-zone" awaiting a command. The communication is broken down as follows:

1. first, the laptop connects to the controller and identifies itself as the client,
2. the controller responds by setting the laptop to the appropriate mode: in this case, the *gesture* mode,
3. once in the *gesture* mode, the laptop continually sends the controller updates, at 3 hertz, on whether or not the robot won the leader election,
4. if a gesture was detected while the robot was selected, then the laptop sends the gesture to the controller,
5. once the gesture is received by the controller, the robot enters the navigation mode, and switches the laptop to the *fiducial* mode, and finally
6. as the robot navigates about the room, if the laptop detects a fiducial, the laptop sends the corresponding information which is used to localize the robot.

For further information on the commands used in our system, refer to appendix A which has a complete list of the RPC methods.

5.3.1 Localization

In order for the robots to drive to and from assigned zones, some form of localization is required. There is a large literature on mobile robot localization (and mapping) [95]. Our robots use a shared global coordinate system to navigate between different zones. Three unique fiducial markers are strategically placed on the walls near each zone. As the robot moves, the fiducials may or may not be visible by the robot. If a fiducial is detected, then a global position is estimated based on the detected size and normal of the fiducial, otherwise, a wheel encoder is used to update the estimate of the robot's position relative to last seen fiducial.

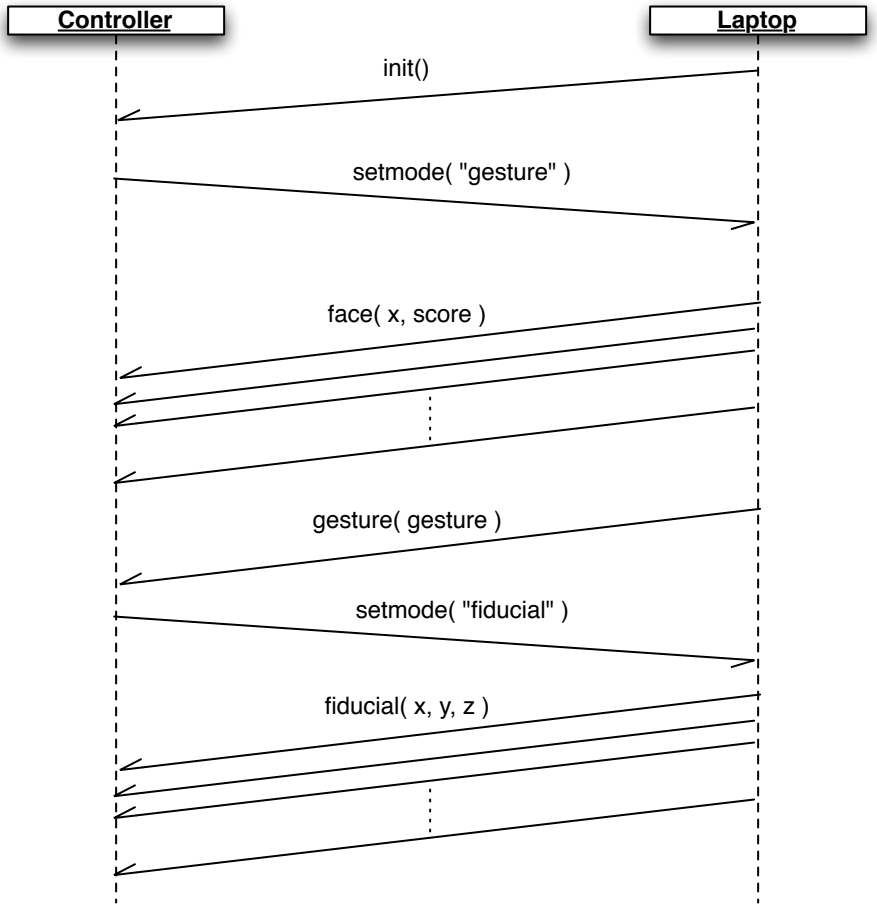


Figure 5.6: Network timing diagram

The robot's odometry, which is calculated by measuring the number of wheel revolutions with the wheel encoders, is used for measuring the robot's position relative to a starting position. Odometry-based localization works well for short-term distances; however, since odometry is computed by integrating motions of movement, small amounts of error introduced at each integration step will accumulate and lead to unusable long-term data. It is therefore critical that the robot's position is periodically corrected; our system corrects odometry errors by resetting the robots position whenever a fiducial marker is detected.

To demonstrate our robot's localization implementation, consider a right-wall-following task. The robot is located in an enclosed rectangle, as seen in Fig 5.7(a), and indefinitely drives parallel to the wall. When an obstacle is detected in front, the robot first turns to the left to avoid the wall, then continues to drive forward. Raw odometry from a single wall-following cycle, displayed in Fig 5.7(b), shows that the wheel-encoder records a smaller angle (< 90 degrees) compared the 90 degree turn the robot actually performed. The position error, most noticeably introduced at *each* turn, accumulates over time and ultimately leads to unusable data, as illustrated in Fig 5.7(c). In the following sections, we will explain how we correct the raw odometry in order to produce a usable robot path as displayed in Fig 5.7(d).

Fiducial markers

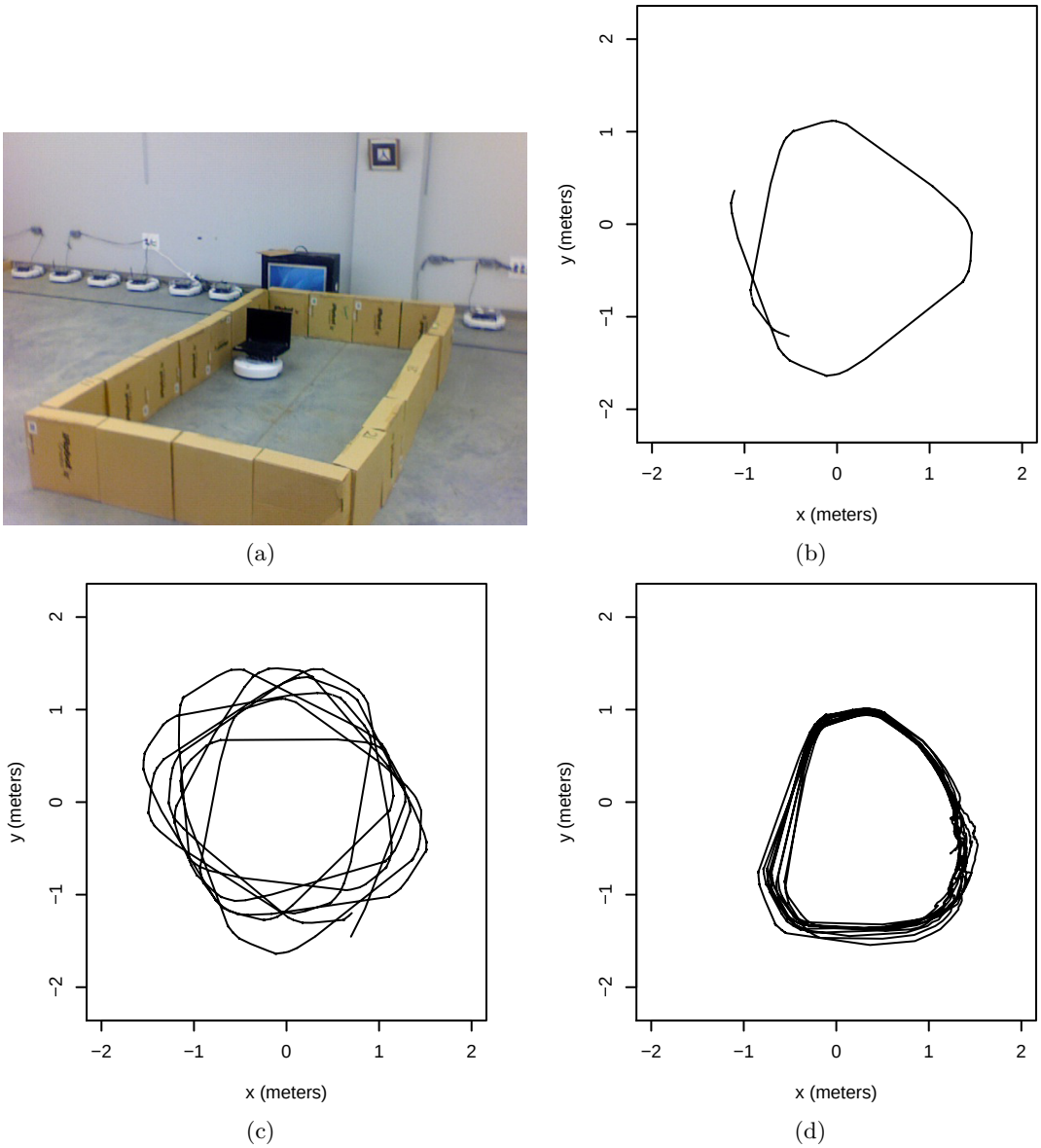
Unique fiduciary markers, shown in Fig 5.8, are mounted on the wall in known locations; The global coordinate and orientation of each fiducial is given to each robot a priori. We use the ARToolkit software library [46], which provides a template matching based fiducial detector. The detector extracts a unique ID corresponding to the detected fiducial, and estimates both the distance to the fiducial and the normal. With the distance, normal, and the fiducial's global pose, which is looked up in a table based on the unique ID, the system is capable of estimating the global pose (x, y, a) of the robot:

$$a = a_{detected} + a_{initial}$$

$$x = x_{detected} cos(a) - y_{detected} sin(a) + x_{initial}$$

$$y = y_{detected} cos(a) + x_{detected} sin(a) + y_{initial}$$

where $x_{detected}$, $y_{detected}$, $a_{detected}$ represent the detected fiducial pose, relative to the robot, and $x_{initial}$, $y_{initial}$, $a_{initial}$ are the components of the fiducial's known global pose.



(a) (b)

(c) (d)

Figure 5.7: The need for odometry correction is illustrated by a (a) small wall-following experiment. (b) The start and end points of a complete cycle inside the boxes do not line up. (c) Overtime, small errors (especially rotational) accumulate in the raw odometry, which is unusable on its own; (d) however, it can be corrected every cycle when a fiducial marker is detected; an extended Kalman filter is used to smooth out errors.

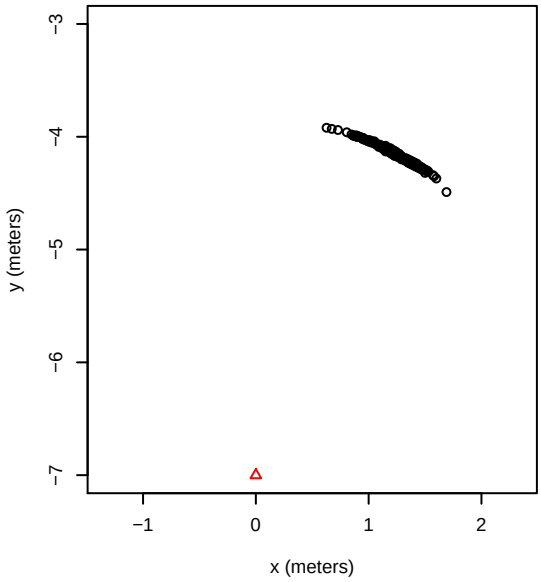
Figure 5.8: A wall-mounted ARtoolkit fiducial used for localization

The robot's orientation is first estimated by looking up the given global orientation of the mounted fiducial. If the fiducial was detected at an angle, i.e. not head on, then the detected angle, which is relative to the global orientation, is added. Once the robot's global orientation is estimated, the detected distance and offset is treated as a vector $(x_{detected}, y_{detected})$ which is rotated about the robot's orientation a. This rotated vector is relative to the fiducial's known global pose, and must therefore be added to the given pose of the fiducial in order to estimate the robot's global pose.

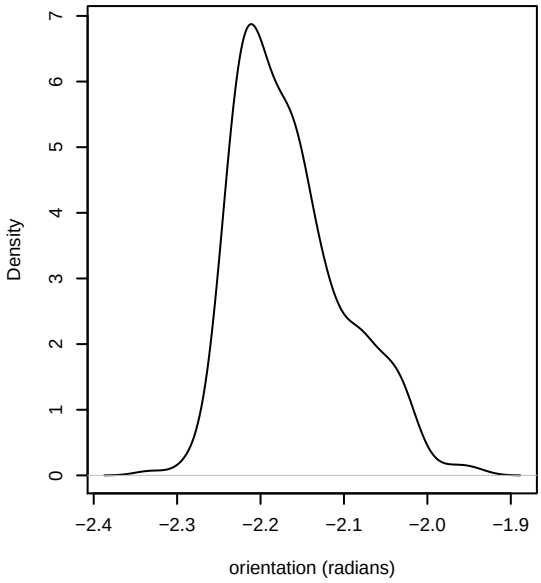
The estimated pose of the robot can be erroneous and may contain noise. Ideally, a perfect fiducial detector would constantly estimate the correct pose without any variance; however, our detector is not perfect and produces many estimates. Fig 5.9(a) shows the estimated pose of a *stationary* robot observing a wall-mounted fiduciary marker: black circles indicate the (x, y) pose of the robot relative to the fiducial, which is indicated by a red triangle. The plot immediately shows that we can not always trust the estimated pose; however, if we assume the estimated pose errors are normally distributed along the arc, then we can safely assume the average estimated pose is likely to be correct. Further investigation of the density distribution of the estimated orientation of the fiducial, plotted in Fig 5.9(b), shows a Gaussian-like distribution.

Kalman filtering

We filter the detected robot pose with a Kalman filter [45]. The Kalman filter is a recursive filter used for estimating the state of a linear dynamic system based on a series of noisy measurements. The filter assumes that all noise in the system is Gaussian and independent,



(a) Pose estimates of a stationary robot (black circles) observing a single wall-mounted fiducial (red triangle) appear to be normally distributed along a circular arc



(b) Density plot of orientation estimates of the fiducial relative to a stationary robot

Figure 5.9: A sample of pose estimates of a stationary robot observing a wall-mounted fiducial marker

and merges all sensor data together by minimizing the mean of squared error.

Our system's fiducial detector appears to have Gaussian noise (as previously shown in Fig 5.9); however, since our fiducial detector can only estimate a pose periodically, when a fiducial is visible, our system is not linear. Therefore we use an extended Kalman filter (EKF) to overcome this limitation. The EKF used in our system was adapted from a sample given in Probabilistic Robotics [96, p. 204].

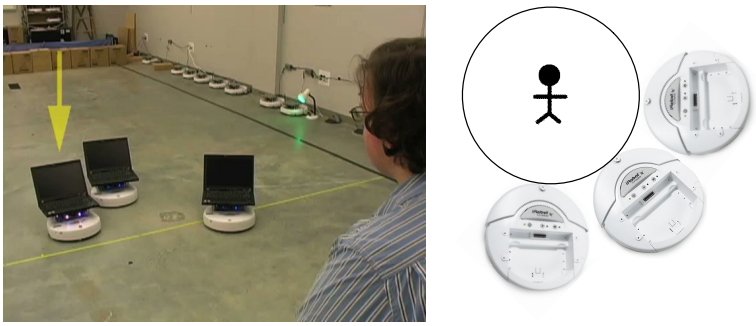
5.3.2 Navigation

Our robots achieve obstacle avoidance with the nearness diagram (ND) algorithm [63]. ND navigates the robot from its current position p to a goal position p' while avoiding both static and dynamic obstacles. The algorithm computes free-walking areas which are safe for robot navigation, by searching for "gaps" between obstacles. Large differences in range sensor readings between contiguous sensors are used to detect a gap, or edges of obstacles. Once all gaps have been detected, "regions" are identified by contiguous gaps. ND then selects an appropriate region which directs the robot towards the goal, but away from obstacles.

Group formation

When the robots approach the user for further instructions, the robots arrange themselves in a circular-arc arrangement around the user. This arrangement, as seen in Fig 5.10(a), provides a direct line of sight between the human and robots to aid task assignment without forcing the user to move about the room. This arrangement is simply formed by having all the robots navigate to a single goal point p'; however, once the robot is 4 meters away from p', the robot stops and considers itself at the goal. The robot then orients itself in the direction of p'. If a stationary robot is already located along the circular arc, then the navigating robot will detect the stationary robot as an obstacle, and will navigate around it. ND will steer the robot around the stationary robot, and will then redirect it to p' once a free-walking region has been found. Eventually the robot will have cleared the obstacle and will halt once it has crossed the circular arc as shown in Fig 5.10(b).

Once the robots have stopped somewhere along the circular arc, 4 meters away from the user, the robots will orient themselves towards p'. If a face is detected, then the robots will pivot in order to center the face in the video camera frame. This ensures that the motion-based gestures will be captured when the user waves his or her hand.



(a) Three robots awaiting instruction in a circular arc formation. The arrow represents the currently selected robot

(b) Robots form a circular arc by driving towards a single point and stopping 4 meters away

Figure 5.10: Robots form a circular arc formation by stopping 4 meters away from a single shared point

5.3.3 Controller summary

Pseudo code giving a high-level summary of the robot controller's navigation, can be found in Algorithm 3. A flowchart diagram in Fig 5.11 illustrates the components of the vision processing system.

5.4 Conclusion

In this chapter, we gave an overview of how the physical robot was put together. We described the layout of our environment the robot was designed for, and gave an overview of a demonstration task which we will use to showcase our system. We also described how the various models fit together to form the robot controller.

Algorithm 3 Robot navigation controller

```
1: if working = true then
2:     if distance to p' < 2 then
3:         working = false
4:         p' ← p_human
5:     else
6:         Drive to goal p' (with ND)
7:     end if
8: else
9:     if distance to p' < 4 then
10:        pivot to center detected face
11:        if gesture = "left" then                ▷ set by the laptop if it won the election
12:            p' ← p_red
13:            working ← true
14:        else if gesture = "right" then
15:            p' ← p_green
16:            working ← true
17:        end if
18:    else
19:        Drive to goal p' (with ND)
20:    end if
21: end if
```

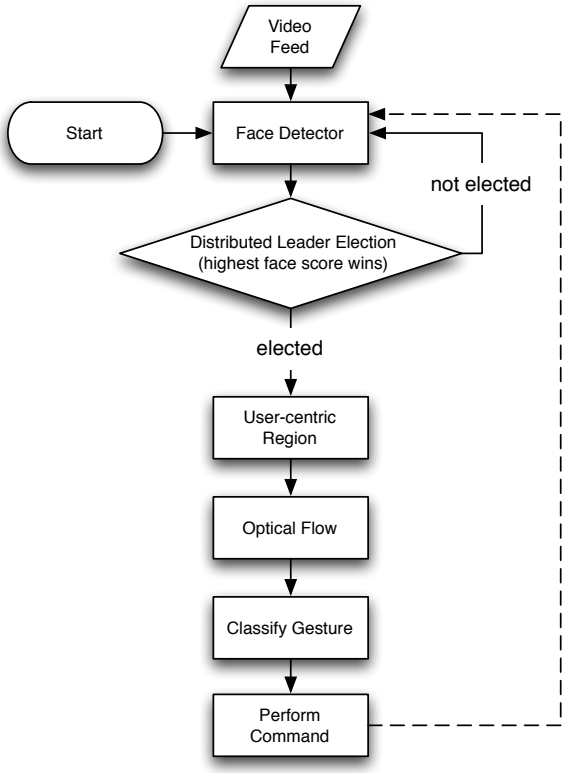


Figure 5.11: Flowchart of the leader election and motion-based gesture recognition process

Chapter 6

Discussion

6.1 Demonstration

We have tested our system with 7 participants who were not involved with the development of the system. Three of the participants were members of the Autonomy Lab, and the other four were SFU students or faculty who were not associated with our lab. A single participant interacted with the robots at any given time. The goal of the demonstration task – to investigate the feasibility of using face engagement and motion-based gestures for commanding an individual robot in a multi-robot system – was explained to each participant. A brief overview of the demonstration task, as described in section 5.2, was orally explained to each participant. The explanation introduced the participant to the human-robot interface and instructed the user to:

1. first select a robot by looking at it, then once the robot started to glow,
2. direct the robot to one of two zones: a green zone, by waving your right hand, or a red zone, by waving your left hand.

Each participant was asked to command two robots to the same zone, and to command the third robot to the other zone. Once the robots reached the zone, they returned to the user. Participants were then encouraged to assign new tasks to returning robots as they saw fit. Some participants waited for all three robots to return before assigning tasks, where as others decided to immediately assign new tasks as each robot arrived. Cases where users waited for all robots to return allowed us to better gauge the effectiveness of face engagement

for robot selection (since the leader election was between three robots). However, since each trial began with all three robots waiting for tasks, we were able to observe each participant selecting a robot from the group of three robots at least once.

Throughout the demonstration, a total number of 77 tasks were assigned to the robots. The next two sections are an informal description and discussion of the performance of the human-robot interface and is split up between robot selection (selecting a particular robot), and task designation (assigning the task to the selected robot). A comprehensive user study is beyond the scope of this thesis.

A three minute video [17] of our demonstration was published in the video track of the 5th annual ACM/IEEE International Conference on Human-Robot Interaction (2010), and is available at `http://www.cs.sfu.ca/~vaughan/movies/hri10.mov`

6.2 Robot selection

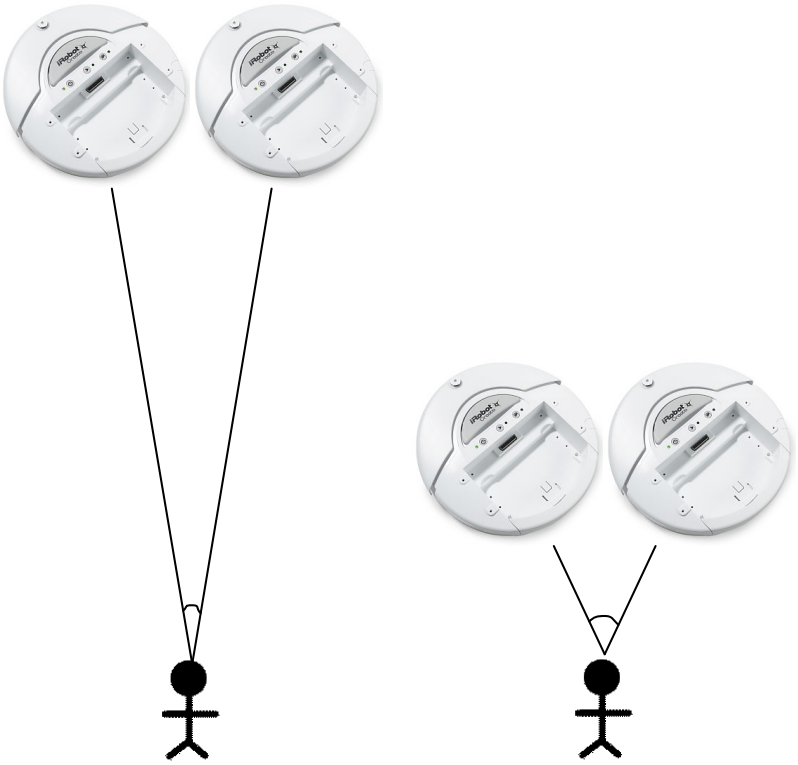
Our leader election algorithm performed as intended: only a single robot ever responded, by glowing, at a given time. In some cases no robots were elected due to equal face scores resulting in a tie. In these cases, participants were encouraged to reposition themselves to break the tie. Ties occurred when two robots awaiting instructions were located very close to each other as illustrated in Fig 6.1(a); however, once the user approached a particular robot, as illustrated in Fig 6.1(b), the angle at which the user had to turn his or her head increased, resulting in a single robot seeing a full frontal face.

In some cases, rather than re-engage one of the two side by side robots, users appeared to be discouraged by a tie and simply tried to select a third robot which they did not originally intend to select.

The face detector implementation provided with OpenCV worked well provided the faces were not too far away from the camera. In our demonstration, the user was at most 4 meters away from the camera and was in a well lit environment. Even with the camera pointing upwards towards the overhead lights, the system was still able to detect faces.

To provide visual feedback, the selected robot would glow with randomly alternating colours. Unfortunately the laptop partly covered the LED which made it hard to see for taller participants. We tried using the laptop's screen as a giant "LED" to provide feedback, but users reported that it was not as satisfying as the glowing LEDs.

Initially, two of the seven participants were unsure which robot was selected and issued



(a) Standing too far from the robots creates a small angle between the line of sight between each robot. Each robot sees a very similar face, resulting in equal face scores

(b) Moving closer to the robots increases the angle the user must turn their head between robots. The robots see different sides of the face: one sees a frontal face, the other sees a profile, thus producing different scores.

Figure 6.1: Varying the distance between the user and two closely located robots can be used to break ties.

commands before any robot had started to glow. However, after we encouraged them to move closer to the robot (and into its video-frame), these two participants were able to command the robots. In other cases, the robots were too close to both the wall and the user, ultimately forcing the user's back up against the wall. These localization-related problems forced the participants to squat down in order to be in the robot's field of view.

In addition to using the laptop to display the same colours as the LEDs, the laptop displayed a copy of the video feed captured by the webcam. This video screen provided the users with visual cues used to help center them in the frame; however, in some trials the screensaver became active and participants could no longer rely on the video-feed. Nonetheless, all participants were able to select their intended individual robot.

6.3 Task designation

Participants then assigned tasks to the selected robot with a motion-based hand gesture. We explicitly demonstrated the two gestures: left hand waving, and right hand waving, hereby referred to as wave-left and wave-right respectively. Using hand waving gestures to assign robots location dependent tasks proved to be challenging in three cases:

1. two of our participants, at first, extended their hands to point left or right rather than wave their hands in a continuous motion,

2. the full hand waving motion of participants who were located too close to the robot, was never captured by the video camera, and

3. the one second optical flow history window required for gesture recognition gave the interface a slow feel.

The accuracy of the system was good: 74 out of 77 commands were correctly executed. The 3 errors occurred when a user issued a command to a robot, and then quickly selected a different robot. This resulted in the newly selected robot classifying the previously issued gesture based on the motion features stored in the optical flow history window. This unintended behaviour could be remedied by reinitializing the optical flow history window whenever a robot is *not* elected.

To avoid classification errors, robots used a high classification threshold. Choosing a high threshold gives a high level of precision, which prevents the robots from incorrectly

classifying a command resulting in opposite behaviour; however, setting the threshold too high limits our level of recall which became an irritant to some participants. After some exposure to the system participants were able to fine tune their gestures to achieve quicker recognition. Providing some sort of feedback mechanism may have decreased the interface's learning curve.

6.4 Navigation

The system occasionally suffered from poor localization issues. The fiducials were difficult for the robots to see from the center of the room, and the odometry of the Creates accumulated error quickly. The poor odometry may have been related to the 5.5 lbs laptop which caused the rear caster to drag whenever the robot turned. Without any ground truth robot position data, it is difficult to truly assess the quality of our navigation; however, the system was usually able to navigate between our purposely large radius zones.

Collisions between robots (sensed by the bumpers), however, were so detrimental, that the robots had to be subsequently repositioned in view of a fiducial before they would localize correctly. Fortunately, collisions were not too common in our system. During the demonstrations, only 4 collisions were observed on our recorded footage. The laptops mounted on the robots hang over the body and bumper of the robot; in one humorous collision, a laptop's DVD eject button was triggered. Partly due to this fact, we decided to limit the driving speed to a slow 0.2 m/s.

Chapter 7

Conclusion

In this thesis, we presented a computer vision-based human-robot interface for selecting and commanding an individual robot from a multi-robot system. A user first selects a robot with face-engagement by simply looking at it. As described in chapter 3, we employed a standard frontal face detector to detect the user's face. The detected-face score of each robot is used in a distributed leader election algorithm to guarantee at most a single robot is selected.

Once a robot has been selected by the user, it can then be commanded by using a motion-based gesture. In chapter 4, we described a real-time classifier which uses motion-cues to discriminate between gestures. Optical flow is cropped in a user-centric region, and classified with a learned adaBoost classifier.

In chapter 5, we described how these two vision components were combined with a customized Create robot. A demonstration task was described to investigate the feasibility of using face engagement and motion-based gestures for commanding an individual robot in a multi-robot system. A discussion of our demonstration in chapter 6 showed that our face-engagement based leader-election could be used to select an individual robot, which could then be commanded to autonomously travel to one of two zones by using motion-based hand gestures.

7.1 Contributions

There has been little work in the human-robot interaction field related to interactions with multi-robot systems. Our vision-based human-robot interface for selecting a particular robot

from a group of robots is a contribution – the first of its kind.

We contribute an observation that it is possible to estimate which camera a user is looking at by comparing the detected-face score from a frontal face detector.

Our final contribution is a real-time motion-based gesture recognition system used for commanding robots.

7.2 Future work

A proper user-study with a larger number of participants would be the next step for evaluating the system; however, the observations so far suggest some useful improvements: the human-robot interface could first be improved by providing a better feedback mechanism. The use of LEDs works well for quickly determining the current robot state; however, it would be valuable to see how users respond to an anthropomorphized robot with eyes. Rather than using glowing LEDs, the robot's "eyes" could be used to show readiness for a command by mimicking bi-directional face engagement (e.g. [74]). This could easily be implemented with virtual eyes on the laptop screen.

Mounting a laptop on the robot was cumbersome. It obscured the robot LEDs and the underlying robot controller software. Velcro is a wonderful technology; however, a better laptop mounting solution (or embedded computer) which does not allow laptop–laptop collisions could be created. Better localization could be achieved using scanning laser rangefinders.

An extension to this system would be to allow users to first select a subset of the robots and then assign that selected group a particular task. The set of gestures could also be extended to allow more dynamic tasks, such as the ability for a user to point to *any* direction or arbitrary place in the environment, and have the robots drive to that location. Detecting pointing gestures and estimating the corresponding vector and intersecting point on the floor has been done for a single robot system (e.g. [52, 60]); however, a challenging task would be to coordinate multiple robots to cooperatively estimate the vector given the system's ability to simultaneously capture images of the user from multiple angles.

In addition to commanding robots with gestures, speech could be used. A user could first select a robot with face-engagement and then utter a command.

7.3 Final comments

Our system provides a proof of concept demonstration that shows it is possible to use face engagement and motion-based hand gestures to select and command an individual robot from a multi-robot system.

As robots become more common in our homes, workplaces, and our society in general, human-robot interactions will become a frequent occurrence. It is therefore important to develop natural and intuitive interfaces to effectively communicate with robots. Face engagement plays a important role in human interactions, and we believe it is a key aspect in creating natural human-robot interactions, as suggested by our system.

Appendix A

RPC methods

This appendix lists the remote procedure calls (RPCs) which are used to communicate between the robot controller, which runs on the Chatterbox's gumstix computer and the laptop, which is responsible for the computer vision related capture and processing. Both controller, and laptop implements a TCP and UDP server; the RPC messages they respond to are documented in the following two sections.

A.1 Controller RPC definitions

`init()`

Arguments

(none)

Return

{} (empty object)

Description

Initialization call made to register laptop's IP with the controller.

`gesture( gesture )`

Arguments

`gesture`	(string) detected gesture

Return

{} (empty object)

Description

Called by the laptop whenever a gesture is detected *and* the robot is elected (i.e. selected by a user).

`fiducial( x, z, angle, id )`

Arguments

`x`	(double) horizontal off-centered location of the fiducial relative to the center of the captured image (in meters)
`z`	(double) distance from camera to fiducial (in meters)
`angle`	(double) orientation of the fiducial relative to camera (in radians)
`id`	(int) fiducial identifier

Return

{} (empty object)

Description

Called by the laptop whenever a fiducial is detected. Robot odometry is corrected based on the estimated position and orientation which are relative to the known position of the fiducial, identified by `id`, which is given a priori.

`face( x, elected, score )`

Arguments

`x`	(double) horizontal off-centered location of the detected face relative to the center of the captured image (in percentage ranging from -0.5 to 0.5)
`elected`	(boolean) whether or not the face is elected
`score`	(int) score of the detected face

Return

{} (empty object)

Description

Called by the laptop whenever a face is detected. If `elected` is true, then the robot glows. The value of `x` is used to pivot the robot in order to center the camera on the face.

A.2 Laptop RPC definitions

`setlight( r, g, b )`

Arguments

`r`	(int) value of the red channel, ranging from 0 to 255
`g`	(int) value of the green channel, ranging from 0 to 255
`b`	(int) value of the blue channel, ranging from 0 to 255

Return

{} (empty object)

Description

Called by the controller to sync up the LED colours of the robot with the colours displayed on the laptop screen.

`join()`

Arguments

(none)

Return

{ `next : (string), tokenid : (string)`}: an object containing the `next` node in the ring, and a `tokenid` used to validate messages passed in the ring.

Description

Called by laptops wishing to join a pre-existing ring. Only laptops which are the owner of a ring will respond to this RPC.

`update( next, tokenid )`

Arguments

`next`	(string) next node in the ring to pass messages to
`tokenid`	(string) an id used to validate messages passed on the ring

Return

{} (empty object)

Description

Called by ring owners whenever a node joins or leaves the ring. Only nodes which are currently connected to a ring respond to messages received only by the appropriate ring owner.

`token( value )`

Arguments

`value`	(election tuple object $\{uid : (string), score : (int)\}$) election tuple used for electing a single robot as the selected robot

Return

{} (empty object)

Description

Called by the previous node in the ring during an election. If a node receives a message containing its own UID, then it is declared the elected node. Otherwise, if the `score` is less than or equal to the nodes score, then it replaces the value of `uid` and `score` with IP address and its own score respectively.

Bibliography

[1] Philip E. Agre and David Chapman. What are plans for? Technical report, Cambridge, MA, USA, 1988.

[2] Isaac Asimov. Runaround. Street & Smith, March 1942.

[3] Shumeet Baluja and Dean Pomerleau. Non-intrusive gaze tracking using artificial neural networks. Technical report, Pittsburgh, PA, USA, 1994.

[4] Shishir Bashyal and Ganesh Kumar Venayagamoorthy. Human swarm interaction for radiation source search and localization. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 1–8, 2008.

[5] Anna Batki, Simon Baron-Cohen, Sally Wheelwright, Jennifer Connellan, and Jag Ahluwalia. Is there an innate gaze module? evidence from human neonates. *Infant Behavior and Development*, 23:223–229, 2000.

[6] Mark Bayazit, Alex Couture-Beil, and Greg Mori. Real-time motion-based gesture recognition using the GPU. In *Proceedings of the IAPR Conference on Machine Vision Applications*, pages 9–12, May 2009.

[7] Mark Becker, Efthimia Kefalea, Eric Mal, Christoph Von, Christoph von der Malsburg, Mike Pagel, Jochen Triesch, Jan C. Vorbrggen, Rolf P. Wrtz, and Stefan Zadel. Gripsee: A gesture-controlled robot for object perception and manipulation. *Autonomous Robots*, 6:203–221, 1999.

[8] Aaron Bobick and James W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):257–267, 2001.

[9] Agnieszka Bojko. Using eye tracking to compare web page designs: A case study. *Journal of Usability Studies*, 1(3):112–120, 2006.

[10] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, Cambridge, MA, 2008.

[11] Cynthia Breazeal. *Designing Sociable Robots*. MIT Press, Cambridge, MA, USA, 2002.

[12] Rechele Brooks and Andrew N. Meltzoff. The development of gaze following and its relation to language. *Developmental Science*, 8(6):535–543, 2005.

[13] Rechele Brooks and Andrew N. Meltzoff. Infant gaze following and pointing predict accelerated vocabulary growth through two years of age : a longitudinal, growth curve modeling study. *Journal of Child Language*, 35(1):207–220, 2008.

[14] Werner. Bux. Token-ring local-area networks and their performance. *Proceedings of the IEEE*, 77(2):238–256, Feb 1989.

[15] Jennifer Casper and Robin Roberson Murphy. Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 33(3):367–385, June 2003.

[16] Ernest J. H. Chang and Rosemary Roberts. An improved algorithm for decentralized extrema-finding in circular configurations of processes. *Communications of the ACM*, 22(5):281–283, may 1979.

[17] Alex Couture-Beil, Richard T. Vaughan, and Greg Mori. Selecting and commanding individual robots in a vision-based multi-robot system. In *HRI '10: Proceedings of the 5th ACM/IEEE international conference on Human robot interaction*, pages 355–356, 2010.

[18] Alex Couture-Beil, Richard T. Vaughan, and Greg Mori. Selecting and commanding individual robots in a vision-based multi-robot system. In *Seventh Canadian Conference on Computer and Robot Vision (CRV)*, pages (to appear – 8 pages), 2010.

[19] Douglas Crockford. The application/json Media Type for JavaScript Object Notation (JSON). RFC 4627 (Informational), July 2006.

[20] Andrew T. Duchowski. A breadth-first survey of eye-tracking applications. *Behavior Research Methods, Instruments, and Computers*, 34(4):455–470, 2002.

[21] Brian R. Duffy. Anthropomorphism and the social robot. *Robotics and Autonomous Systems*, 42(3–4):177–190, 2003.

[22] Alexei A. Efros, Alexander C. Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *Proc. 9th Int. Conf. Computer Vision*, volume 2, pages 726–733, 2003.

[23] Teresa Farroni, Gergely Csibra, Francesca Simion, and Mark H. Johnson. Eye contact detection in humans from birth. In *Proceedings of the National Academy of Sciences of the United States of America*, pages 9602–9605, 1999.

[24] Teresa Farroni, Mark H. Johnson, and Gergely Csibra. Mechanisms of eye gaze perception during infancy. *Journal of Cognitive Neuroscience*, 16(8):1320–1326, 2004.

[25] Alireza Fathi and Greg Mori. Action recognition by learning mid-level motion features. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[26] Frank J Ferrin. Survey of helmet tracking technologies. In *SPIE Proceedings Vol. 1456 Large-Screen Projection, Avionic, and Helmet-Mounted Displays*, pages 86 – 94, 1991.

[27] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting,. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997.

[28] Masahiro Fujita and Hiroaki Kitano. Development of an autonomous quadruped robot for robot entertainment. *Autonomous Robots*, 5(1):7–18, 1998.

[29] Susan R. Fussell, Sara Kiesler, Leslie D. Setlock, and Victoria Yew. How people anthropomorphize robots. In *HRI '08: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 145–152, New York, NY, USA, 2008. ACM.

[30] Brian P. Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, sep 2004.

[31] Erving Goffman. *Behavior in Public Places: Notes on the Social Organization of Gatherings*. Free Press, September 1966.

[32] Joseph H. Goldberg and Xerxes P. Kotval. Computer interface evaluation using eye movements: methods and constructs. *International Journal of Industrial Ergonomics*, 24(6):631–645, October 1999.

[33] Michael A. Goodrich and Alan C. Schultz. Human-robot interaction: a survey. *Foundations and Trends in Human-Computer Interaction*, 1(3):203–275, 2007.

[34] JSON-RPC Working Group. Json-rpc 2.0 specification proposal. `http://json-rpc.org/wiki/specification`, 2005. Retrieved January 17, 2010.

[35] Nicolas Guèguen and Cèline Jacob. Direct look versus evasive glance and compliance with a request. *The Journal of Social Psychology*, 142(3):393–396, 2002.

[36] B.K.P. Horn and B.G. Schunk. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[37] Helge Huttenrauch, Anders Green, Mikael Norman, Lars Oestreicher, and Kerstin Severinson Eklundh. Involving users in the design of a mobile office robot. In *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, pages 113–124, 2004.

[38] Tsukasa Ike, Nobuhisa Kishikawa, and Bjorn Stenger. A real-time hand gesture interface implemented on a multi-core processor. In *IAPR Conference on Machine Vision Applications*, pages 9–12, 2007.

[39] Michita. Imai, Takayuki. Kanda, Tetsuo. Ono, Hiroshi. Ishiguro, and Kenji. Mase. Robot mediated round table: Analysis of the effect of robot's gaze. In *Robot and Human Interactive Communication, 2002. Proceedings. 11th IEEE International Workshop on*, pages 411–416, 2002.

[40] Michita Imai, Tetsuo Ono, and Hiroshi Ishiguro. Robovie: Communication technologies for a social robot. *International Journal of Artificial Life and Robotics*, 6:73–77, 2003.

[41] Hiroshi Ishiguro and Shuichi Nishio. Building artificial humans to understand humans. *Journal of Artificial Organs*, 10(3):133–142, September 2007.

[42] Hiroshi Ishiguro, Tetsuo Ono, Michita Imai, Takeshi Maeda, Takayuki Kanda, and Ryohei Nakatsu. Robovie: an interactive humanoid robot. *Industrial robot: An international journal*, 28(6):498–503, 2001.

[43] Robert J. K. Jacob. The use of eye movements in human-computer interaction techniques: what you look at is what you get. *ACM Transactions on Information Systems*, 9(2):152–169, 1991.

[44] Robert. J. K. Jacob and Keith. S. Karn. Eye tracking in human-computer interaction and usability research: Ready to deliver the promises. *The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research*, pages 573–603, 2003.

[45] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME – Journal of Basic Engineering*, (82 (Series D)):35–45, 1960.

[46] Hirokazu Kato and Mark Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, page 85, Washington, DC, USA, 1999. IEEE Computer Society.

[47] Jun Kato, Daisuke Sakamoto, Masahiko Inami, and Takeo Igarashi. Multi-touch interface for controlling multiple mobile robots. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 3443–3448, New York, NY, USA, 2009. ACM.

[48] Adam Kendon. Some functions of gaze-direction in social interaction. *Acta psychologica*, 26:22–63, 1967.

[49] Adam Kendon. Gesture. *Annual Review of Anthropology*, 26(1):109–128, 1997.

[50] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, and Hitoshi Matsubara. Robocup: A challenge problem for ai and robotics. In *RoboCup-97: Robot Soccer World Cup I*, pages 1–19, 1998.

[51] Chris L. Kleinke. Gaze and eye contact: A research review. *Psychological Bulletin*, 100(1):78–100, 1986.

[52] David Kortenkamp, Eric Huber, R. Peter Bonasso, and Metrica Inc. Recognizing and interpreting gestures on a mobile robot. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 915–921. AAAI Press/The MIT Press, 1996.

[53] Hideki Kozima, Marek P. Michalowski, and Cocoro Nakagawa. Keepon: A playful robot for research, therapy, and entertainment. *International Journal of Social Robotics*, 1(1):3–18, 2009.

[54] Yoshinori Kuno, Michie Kawashima, Keiichi Yamazaki, and Akiko Yamazaki. Importance of vision in human-robot communication understanding speech using robot vision and demonstrating proper actions to human vision. In *Intelligent Environments*, Advanced Information and Knowledge Processing, pages 183–202. Springer London, 2009.

[55] Gerald. L. Lohse. Consumer eye movement patterns on yellow pages advertising. *Journal of Advertising*, 26(1):61–73, 1997.

[56] Matthew M. Loper, Nathan P. Koenig, Sonia H. Chernova, Chris V. Jones, and Odest C. Jenkins. Mobile human-robot teaming with environmental tolerance. In *HRI '09: Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 157–164, New York, NY, USA, 2009. ACM.

[57] Päivi Majaranta and Kari-Jouko Räihä. Twenty years of eye typing: systems and design issues. In *ETRA '02: Proceedings of the 2002 symposium on Eye tracking research & applications*, pages 15–22, New York, NY, USA, 2002. ACM.

[58] S.G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:674–693, 1989.

[59] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *An Introduction to Information Retrieval*. 2008.

[60] Christian Martin, Frank-Florian Steege, and Horst-Michael Gross. Estimation of pointing poses for visual instructing mobile robots under real-world conditions. *Robotics and Autonomous Systems*, 57:(to appear), 2009.

[61] Maja J. Matarić. Designing emergent behaviors: From local interactions to collective intelligence. In *Proceedings of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, volume 2, pages 432–441, 1992.

[62] James McLurkin, Jennifer Smith, James Frankel, David Sotkowitz, David Blau, and Brian Schmidt. Speaking swarmish: Human-Robot interface design for large swarms of autonomous mobile robots. In *Proceedings of the AAAI Spring Symposium*, 2006.

[63] Javier Minguez and Luis Montano. Nearness diagram (nd) navigation: Collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation*, 20:2004, 2004.

[64] Sushmita Mitra and Tinku Acharya. Gesture recognition: A survey. *IEEE Transactions On Systems, Man, And Cybernetics - Part C: Applications And Reviews*, 37(3):311–324, May 2007.

[65] Thomas B. Moeslund, Adrian Hilton, and Volker Kruger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104:90–126, 2006.

[66] Carlos H. Morimoto and Marcio R.M. Mimica. Eye gaze tracking techniques for interactive applications. *Computer Vision and Image Understanding*, 98(1):4–24, 2005.

[67] Bilge Mutlu and Jodi Forlizzi. Robots in organizations: the role of workflow, social, and environmental factors in human-robot interaction. In *HRI '08: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 287–294, New York, NY, USA, 2008. ACM.

[68] Bilge Mutlu, Toshiyuki Shiwa, Takayuki Kanda, Hiroshi Ishiguro, and Norihiro Hagita. Footing in human-robot conversations: how robots might shape participant roles using gaze cues. In *HRI '09: Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 61–68, New York, NY, USA, 2009. ACM.

[69] Bilge Mutlu, Fumitaka Yamaoka, Takayuki Kanda, Hiroshi Ishiguro, and Norihiro Hagita. Nonverbal leakage in robots: communication of intentions through seemingly unintentional behavior. In *HRI '09: Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 69–76, New York, NY, USA, 2009. ACM.

[70] Amir M. Naghsh, Jeremi Gancet, Andry Tanoto, and Chris Roast. Analysis and design of human-robot swarm interaction in firefighting. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN'08)*, pages 255–260, 2008.

[71] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with CUDA. *Queue*, 6(2):40–53, 2008.

[72] N. J. Nilsson. Shakey the robot. Technical Report 323, 1984.

[73] Dan R. Olsen, Jr. and Stephen Bart Wood. Fan-out: measuring human control of multiple robots. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 231–238, New York, NY, USA, 2004. ACM.

[74] Hirotaka Osawa, Ren Ohmura, and Michita Imai. Anthropomorphization method using attachable humanoid parts. In *HRI '09: Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 207–208, New York, NY, USA, 2009. ACM.

[75] Constantine P. Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *Computer Vision, 1998. Sixth International Conference on*, pages 555–562, 1998.

[76] Lynne E. Parker. Distributed intelligence: Overview of the field and its application in multi-robot systems. *Journal of Physical Agents*, 2(1):5–14, 2008.

[77] David Payton. Pheromone robotics. `http://www.swarm-robotics.org/SAB04/presentations/payton-review.pdf`, 2004. Slides from a presentation at the Swarm Robotics Workshop, SAB04. Retrieved September 28, 2009.

[78] David Payton, Mike Daily, Regina Estowski, Mike Howard, and Craig Lee. Pheromone robotics. *Autonomous Robots*, 11(3):319–324, 2001.

[79] Dennis Perzanowski, Alan C. Schultz, William Adams, Magda Bugajska, Elaine Marsh, Greg Trafton, Derek Brock, Marjorie Skubic, and Myriam Abramson. Communicating with teams of cooperative robots. In *Proceedings from the 2002 NRL Workshop on Multi-Robot Systems*, pages 16–20. Kluwer, 2002.

[80] Dennis Perzanowski, Alan C. Schultz, William Adams, Elaine Marsh, and Magda Bugajska. Building a multimodal human-robot interface. *Intelligent Systems, IEEE*, 16(1):16–21, Jan-Feb 2001.

[81] Michael Yu. Rachkov, Lino Marques, and Aníbal T. Almeida. Multisensor demining robot. *Autonomous Robots*, 18(3):275–291, 2005.

[82] Marie L. Radford. Approach or avoidance? the role of nonverbal communication in the academic library user's decision to initiate a reference encounter. *Library Trends*, 46(4):699–717, 1998.

[83] Cen Rao, Alper Yilmaz, and Mubarak Shah. View-invariant representation and recognition of actions. *Int. Journal of Computer Vision*, 50(2):203–226, 2002.

[84] D. A. Robinson. A method of measuring eye movements using a scleral search coil in a magnetic field. *IEEE transactions on bio-medical engineering*, 10:137–145, 1963.

[85] O. Rogalla, M. Ehrenmann, R. Zllner, R. Becher, and R. Dillmann. Using gesture and speech control for commanding a robot assistant. In *IEEE International Workshop*

on Robot and Human Interactive Communication, Piscataway, pages 454–459. IEEE Press, 2002.

[86] Brian Scassellati. Investigating models of social development using a humanoid robot. In Barbara Webb and Thomas Consi, editors, *Biorobotics*. MIT Press, 2001.

[87] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[88] Eli Shechtman and Michal Irani. Space-time behavior based correlation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[89] Mark Snyder, John Grather, and Kristine Keller. Staring and compliance: A field experiment on hitchhiking. *Journal of Applied Social Psychology*, 4(2):165–170, 1974.

[90] S. W. Squyres, R. E. Arvidson, J. F. Bell, J. Bruckner, N. A. Cabrol, W. Calvin, M. H. Carr, P. R. Christensen, B. C. Clark, L. Crumpter, D. J. Des Marais, C. d'Uston, T. Economou, J. Farmer, W. Farrand, W. Folkner, M. Golombek, S. Gorevan, J. A. Grant, R. Greeley, J. Grotzinger, L. Haskin, K. E. Herkenhoff, S. Hviid, J. Johnson, G. Klingelhofer, A. H. Knoll, G. Landis, M. Lemmon, R. Li, M. B. Madsen, M. C. Malin, S. M. McLennan, H. Y. McSween, D. W. Ming, J. Moersch, R. V. Morris, T. Parker, J. W. Rice, L. Richter, R. Rieder, M. Sims, M. Smith, P. Smith, L. A. Soderblom, R. Sutlivan, H. Wanke, T. Wdowiak, M. Wolff, and A. Yen. The Opportunity rover's Athena science investigation at Meridiani Planum, Mars. *Science*, 306(5702):1698–1703, 2004.

[91] India Starker and Richard A. Bolt. A gaze-responsive self-disclosing display. In *CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 3–10, New York, NY, USA, 1990. ACM.

[92] Maria Staudte and Matthew W. Crocker. Visual attention in spoken human-robot interaction. In *HRI '09: Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 77–84, New York, NY, USA, 2009. ACM.

[93] E. Svensson, M. Angelborg-Thanderz, L. Sjoberg, and S. Olsson. Information complexity – mental workload and performance in combat aircraft. *Ergonomics*, 40(3):362–380, 1997.

[94] Andrew S. Tanenbaum. *Modern Operating Systems*. Prentice Hall, 2001.

[95] Sebastian Thrun. Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[96] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, September 2005.

[97] Jochen Triesch, Christof Teuscher, Gedeon O. Deak, and Eric Carlson. Gaze following: why (not) learn it? *Developmental Science*, 9(2):125–147, March 2006.

[98] Richard T. Vaughan. Massively multi-robot simulation in Stage. *Swarm Intelligence*, 2(2):189–208, December 2008.

[99] Paul Viola and Michael Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[100] Kazuyoshi Wada, Takanori Shibata, Tomoko Saito, and Kazuo Tanie. Effects of robot-assisted activity for elderly people and nurses at a day service center. *Proceedings of the IEEE*, 92(11):1780–1788, Nov. 2004.

[101] Stefan Waldherr, Roseli Romero, and Sebastian Thrun. A gesture based interface for human-robot interaction. *Autonomous Robots*, 9:151–173, 2000.

[102] Hirotake Yamazoe, Akira Utsumi, Tomoko Yonezawa, and Shinji Abe. Remote gaze estimation with a single camera based on facial-feature tracking without special calibration actions. In *ETRA '08: Proceedings of the 2008 symposium on Eye tracking research & applications*, pages 245–250, New York, NY, USA, 2008. ACM.

[103] Ming-Hsuan Yang and Narendra Ahuja. *Face Detection And Gesture Recognition For Human-Computer Interaction*. Kluwer Academic Publishers, 2001.

[104] Ruigang Yang and Zhengyou Zhang. Eye gaze correction with stereovision for video-teleconferencing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7):956–960, 2004.

[105] Tomoko Yonezawa, Hirotake Yamazoe, Akira Utsumi, and Shinji Abe. Evaluating crossmodal awareness of daily-partner robot to user's behaviors with gaze and utterance detection. In *Casemans '09: Proceedings of the 3rd ACM International Workshop on Context-Awareness for Self-Managing Systems*, pages 1–8, New York, NY, USA, 2009. ACM.

[106] Gregory S. Young, Noah Merin, Sally J. Rogers, and Sally Ozonoff. Gaze behavior and affect at 6 months: predicting clinical outcomes and language development in typically developing infants and infants at risk for autism. *Developmental Science*, 12(5):798–814, 2009.

[107] Shumin Zhai, Carlos Morimoto, and Steven Ihde. Manual and gaze input cascaded (MAGIC) pointing. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 246–253, New York, NY, USA, 1999. ACM.

[108] Shengdong Zhao, Koichi Nakamura, Kentaro Ishii, and Takeo Igarashi. Magic cards: a paper tag interface for implicit robot control. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 173–182, New York, NY, USA, 2009. ACM.