CuWITH: A Curiosity driven Robot for Office Environmental Security

Sean William Gordon

A thesis submitted in partial fulfilment for the degree of Master of Computer and Information Science

October 1, 2009

School of Computing and Mathematical Sciences

Primary Supervisor: Dr. Paul S. Pang

Secondary Supervisor: Prof. Nikola Kasabov

ABSTRACT

The protection of assets is an important part of daily life. Currently this is done using a combination of passive security cameras and security officers actively patrolling the premises. However, security officers, being human, are subject to a number of limitations both physical and mental. A security robot would not suffer from these limitations, however currently there are a number of challenges to implementing such a robot. These challenges include navigation in a complex real-world environment, fast and accurate threat detection and threat tracking. Overcoming these challenges is the focus of my research.

To that end a small security robot, the CuWITH or Curious WITH,has been developed and is presented in this thesis. The CuWITH utilises a programmable navigation system, curiosity-based threat detection and curiosity-driven threat tracking curiosity to protect a real office environment.

In this thesis we will first discuss the CuWITH's system design in detail, with a particular focus on the components and the architectural strategies employed. We then move to a more detailed examination of the mathematical underpinnings of the CuWITHs curiosity based threat detection and curiosity driven threat tracking. The details of the CuWITH's navigation will also be explained.

We will then present a number of experiments which demonstrate the effectiveness of the CuWITH. We show that the programmable navigation of the CuWITH, although simple, allows for easy modification of the patrol path without risk to the stability of the system. We will then present the results of both offline and online testing of the CuWITH's curiosity based threat detection. The reaction time and accuracy of the CuWITHs curiosity driven threat tracking will also be illustrated. As a final test the CuWITH is instructed to execute a patrol in a real office environment, with threatening and non-threatening persons present. The results of this

test demonstrate all major systems of the CuWITH working together very well and successfully executing the patrol even when moved to a different environment.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF PAPERS AND DEMONSTRATIONS

- Sean W. Gordon, Shaoning Pang, Ryota Nishioka, Nikola Kasabov and Takeshi Yamakawa, Vision Based Mobile Robot for Indoor Environmental Security,ICONIP 2008, Part 1, LNCS 5506, pp. 959-966, Springer-Verlag Berlin Heidelberg 2009
- Demonstration at the ICONIP conference 2008, The CuWITH successfully completed several security patrols over a hour.
- Demonstrated on the Sunrise program on TV3. Online `http://tinyurl.com/cn3mfk`, last accessed 7th May 2009

ATTESTATION OF AUTHORSHIP

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning

Sean William Gordon

ACKNOWLEDGEMENTS

I am grateful to my supervisor Dr. Paul Pang for his constant help and support during this research. Cooperation with him was highly motivating and I benefited greatly from his careful advice.

I'd like to thank Kitakyushu Institute of Technology, Japan, for generously supporting this research, in particular by supplying the WITH robot. I especially want to thank Ryota Nishioka for the time and effort he spent helping to develop the CuWITH, and for his patience in answering my questions on the WITH.

I would also like to thank Gary Chen for his help in developing the threat detection of the CuWITH. I thank Stefan Schliebs for his insightful advice on research and technical writing. In addition I would like to thank all KEDRI staff especially Prof. Nikola Kasabov, Joyce D'Mello and other staff who were always there to help me complete my research.

My thanks also go to the many people in and around the KEDRI offices who lent me their faces during this research, I could not have done this without their help.

My special thanks goes to my family. The relaxation time I could spend with them and the support they gave me has been essential during this research.

1. INTRODUCTION

1.1 Security in an Office Environment

Security is an important factor in our everyday lives, especially when it comes to the security of an office environment such as a bank. An office can hold valuable property such as desktop computers and laptops. The loss of such items can cost the business time and money. More importantly, offices also commonly hold confidential client information. If this is stolen, it can cost businesses the trust (and custom) of clients. The protection of such areas is therefore extremely important, requiring a security system which can reliably protect an office 24 hours a day, 7 days a week.

For security in a office environment, two types of security systems are popularly used: stationary security camera monitoring and patrolling mobile security agents. Security cameras are passive security agents that are often installed in key locations of the office. These cameras are then constantly monitored in a security room. If an intruder is seen through one of the security cameras an alarm will be raised. Mobile security agents, on the other hand, are active security agents that regularly patrol the office searching for threats. Normally human security guards are used in this role. If during their patrol they detect an intruder, they can take immediate action to apprehend the intruder. Each system has its advantages, security cameras can cover a large area of the office, while a security agent is limited to their immediate surroundings. However, security cameras suffer from blind spots, areas of the office which are not in the line of sight of any camera. Furthermore, unlike mobile security agents, security cameras cannot immediately take action to apprehend an intruder.

In the presented research, mobile security agents are of particular interest. They are required to regularly patrol the office, actively search for threats and act intelligently and quickly if a threat is detected. Human security guards satisfy these

requirements, however they have several limitations:

- Human security guards can suffer from physiological conditions such as fatigue and illness. Both can result in the security guard being less alert, decrease their stamina and strength and (in the case of fatigue) can negatively affect their mental acuity, resulting in poor decision making.

- The quality of the service that a human security guard provides can vary greatly both between different individuals and between one day and the next for a single individual

- Human security guards can also be corrupt and compromise the safety of the office for their own personal gain.

- Human security guards have limited threat detection capabilities, restricted to their sense of sight and hearing. The sense of sight in particular is limited, its effectiveness is severely reduced in low-light conditions and humans cannot see beyond a specific viewing arc. Equipment can be developed to overcome this issue, however this equipment is often either expensive or difficult to use.

Although today's robot technology is not advanced enough, yet, to replace human security guards, mobile security robots have the potential to overcome some of these difficulties. A robot will not get sick or fatigued. It can be relied on to provide the same quality of security at all times and cannot be tempted to compromise the offices security for personal gain. Security robots can also be upgraded to utilise a variety of different sensors, including infra-red and omnidirectional cameras.

1.2 Challenges for Developing a Mobile Security Robot

When developing a mobile security robot, there are three major challenges which must be addressed. These are security patrol navigation, threat detection and threat tracking.

1.2.1 Security Patrol Navigation

Navigation in an office environment is a complex challenge. Objects (such as tables) can have unusual shapes and have their locations and orientation modified. The structure of the office can include a variety of different rooms of different shapes and sizes. The robot often will suffer from undesirable deviations in its path due to systematic errors (such as one wheel being larger than the rest) and non-systematic errors (such as the wheel slipping) (Borenstein, Everett, Feng, & Wehe, 1997; Filliat & Meyer, 2003). Despite these confounding factors a security robot must be able to patrol this environment regularly and reliably. The most obvious issue is that of localisation: how can the robot determine where it is located in the office? There are a number of methods that can be used, including odometry, landmark navigation, active beacons and map matching (Borenstein et al., 1997).

Odometry is the simplest of these methods. If the robot knows its starting location and heading, it can estimate where it is (relative to the starting location) at any time by integrating its velocity with respect to time. This localisation method is very simple, fast and has been applied successfully in the past (Chenavier & Crowley, 1992), however it is particularly sensitive to systematic and non-systematic errors (Filliat & Meyer, 2003; Borenstein et al., 1997). As a result it is only accurate over short periods of time, eventually the error accumulates and renders the odometry position meaningless.

Landmark navigation is another method of localization that uses distinct features in the environment to determine the robots location. This can take a variety of forms, for example a yellow square may be placed at a specific position on a wall. If the robot detects this square, it can can match it to a location on a map, and so the robot can determine its approximate position. This has been used successfully in security robots that have been developed previously (Shimosasa et al., 2000), however the use of this method requires landmarks to be placed in the environment. This makes moving the robot to a different environment more difficult and costly, also it is preferable that the environment does not need to be modified for the security robot to navigate.

The use of active beacons is a method of localization which has been used to localize security robots (Kim, Kim, Lee, & Lee, 2006; Borenstein et al., 1997). It involves using several transmitters in the environment to determine the robots location. If the transmitters are located at unique, fixed locations, the robot can determine its position relative to the transmitters. This method of localization is reliable and accurate, however it suffers from the same issues as landmark navigation, namely the difficultly and cost of installing the beacons, and that the environment needs to be modified.

Map matching is a more complex localization method. It is a two step process, the robot build a map of its current surroundings using its sensors, then attempts to match this local map with a global map of the environment. If a match is found, the robots location has been determined. Map matching has an advantage over landmark navigation and active beacons in that it does not require the environment to be modified. However matching the local map to the global map is difficult task. The local map must be extremely accurate, and if the environment is dynamic or has a lack of easily distinguishable features map matching can fail. Map matching has been successfully used for robot navigation in simulation (Rencken, 1993) and on a real robot (Thrun, 1998; Lee, Chung, & Kim, 2003).

1.2.2 Threat Detection

Threat detection is the core of any security robot. There are several different techniques that can be used to detect threats to the office that the robot is protecting.

The simplest approach is to detect any people in the office (often using an infrared sensor or camera). If a person is detected, an alarm is raised. This technique cannot work unless there is a guarantee that there should be no people present in the environment - in this situation, any person detected in the environment can be confirmed as an intruder. However if this can be guaranteed (for example, if the office closes between the hours of 5.30PM and 8.30AM) then, because it is so simple, this method of threat detection is robust and reliable. A number of security robots use this technique (Chien, Su, & Guo, 2005; Luo, Lin, Chen, & Su, 2006; Shimosasa

et al., 2000). However, even given this, a security robot that relies on such simple threat detection technique cannot provide protection to an office while there are people working in it. This is a severe limitation.

A more advanced threat detection method is face recognition. In this method, the robot is trained to recognize a number of people that are allowed in the office. If the robot detects a face during its security patrol, it will immediately attempt to recognize this face. If it fails, then the face is classified as a threat and the robot will take some corrective action. Face recognition can be extremely reliable, robust to both the lighting and pose of the face (Liu, Wang, & Feng, 2005). However such an accurate face recognition system is very computationally demanding, and may require powerful external computers to perform in real time (Liu et al., 2005). Face recognition can be performed using more limited computational resources, however these methods are often sensitive to pose, lighting and the distance the face is from the camera (Treptow, Cielniak, & Duckett, 2005; Zhang, Yan, & Lades, 1997; Turk & Pentland, 1991). Another limitation of face recognition is that often it relies on accurate and fast face extraction. Errors can be introduced if the method suffers from false positives (detecting a non-existent face), and if it cannot crop the face accurately (i.e. if parts of the background are include with the face data). The former can cause the robot to raise an alarm for threats that are non-existent, the latter can deteriorate the reliability of the face recognition. Furthermore, threat detection based on face recognition is limited because it is so reliant on faces. If the intruders face is not visible, face recognition is not effective as a threat detection strategy. Also, an intruder can make changes to the environment (they could break a door down, for example) which can reliably indicate the safety of the office has been compromised. A threat detection system that relies on face recognition will not detect such changes, and is therefore easier to evade.

To overcome the above limitation, a new method for threat detection is environmental change detection. This involves the robot learning what the state of the office should be (the ideal state), and determining if there is a significant difference between current state and the ideal state. If there is a significant difference then

the security of the environment has been compromised. This method has been implemented in an autonomous security robot (Andreasson, Magnusson, & Lilienthal, 2007) and was capable of detecting both spatial changes and changes in colour. The limitation of this technique is that it will detect any change in the environment. If the office is static and there no people in the environment, this could work, however if this is not the case then a threat detection system using change detection faces the additional challenge of determining what is a benign change, and what is a change that indicates a threat to the office. This is not trivial, for example separating people that are allowed in the office from people that are not will require the robot to recognize a person from a wide range of angles, poses, lighting conditions, and possibly even different clothing and hair style. Furthermore as the complexity increases the computational complexity will also increase, this may mean that the security robot is unable to perform threat detection autonomously, as it does not have the computational power to execute the threat detection in real time.

1.2.3 Threat Tracking

Whenever a threat is detected, a security robot is required to immediately take some corrective actions. A simple action is to start immediately some kind of security alarm to raise alert of people, since security robot has recorded on video any threat that has been detected, as evidence against the intruder (Shimosasa et al., 2000). There are also robots that, after detecting something which may be a threat, advance towards the threat to get a clearer view (Treptow et al., 2005). However threat tracking for security is not a topic which has been studied extensively in the literature.

1.3 Objectives of this Research

The topic of security robots is an extremely broad one. Rather than attempt to cover the entirety of this field, in this masters thesis we choose to focus on threat detection and a simple security action. To that end a method of threat detection is presented in this thesis which uses curiosity modelling to detect threats. The

concept of curiosity is that as the robot patrols the office, it can expect to encounter certain objects (e.g. particular people allowed in the office). If an intruder enters the environment, they will be outside the robot expectations and so can be regarded as curious to the robot. This is not necessarily limited to people, this curiosity can also extend to changes that an intruder can make while in the office (e.g. broken doors or windows). Because this curiosity based threat detection is not attempting to recognize individuals it is potentially faster and more reliable than threat detection methods based on face recognition. Furthermore by focusing specifically on unusual objects it avoids some of the pitfalls of change detection (sensitivity to insignificant changes). To complement this curiosity based threat detection, a security action technique has been developed. This technique uses the curiosity of the robot to drive its actions, causing it to actively move to investigate and track any threats that it detects. Considering the above requirements the objectives of this research are:

- to develop a threat detection method using curiosity modelling that utilises the difference between what the robot expects to encounter in the office and what it actually encounters to compute a curiosity value. This curiosity value is used to determine if a threat is present in the area.

- to implement a curiosity driven technique for tracking and following threats that have been detected.

- to design and build a small, proof-of-concept autonomous security robot - the Curious WITH or CuWITH - which can be used as an experimental platform to test the effectiveness of curiosity modelling for threat detection and curiosity driven behaviour for security action. CuWITH is implemented using the WITH robot developed by Kitakyushu Institute of Technology in Japan (Mori, Sato, Sonoda, & Ishii, 2007).

1.4 Organization of the Thesis

This Thesis is organized into the following chapters:

Chapter 2 provides a literature review on previous research into security robots and curiosity modelling. In the review of security robots several are considered, including robots developed by Liu et al. (2005), Treptow et al. (2005) and Andreasson et al. (2007). Curiosity Modelling is split into three categories; models based on uncertainty, models based on prediction error and curiosity based on multi-agent cILDA.

Chapter 3 discusses the system architecture of the developed CuWITH, including the programmable patrol navigation, curiosity based threat detection (CBTD) and the curiosity driven threat tracking. The execution procedure is also explained, with a particular focus on how the CuWITH's two major threads (the navigation thread and the main thread) synchronise and exchange information.

Chapter 4 explains in detail the proposed threat detection method (based on curiosity modelling) and the curiosity driven technique used for threat tracking. The implementation of these methods on the CuWITH are also described in detail at the end of the chapter.

Chapter 5 presents several ancillary components that, while not the focus of this thesis, are necessary to implement the CuWITH. This includes the programmable patrol navigation, object extraction and camera monitoring components.

Chapter 6 gives the experiments used to investigate the effectiveness of the CuWITH in protecting an office environment. The three most important components of the CuWITH (the programmable navigation, CBTD and curiosity driven threat tracking) are tested separately. A unified system test is also performed, At the end of the chapter an evaluation of the CuWITH system as a whole is then presented.

Finally Chapter 7 presents the conclusions of this study as well as directions for future work.

2. LITERATURE REVIEW AND MOTIVATION

There is a scarcity of research on utilising curiosity to enhance the capabilities of a security robot. However there is a considerable body of research on the implementation of a security robot and on curiosity modelling. There are several security robots that have been presented in the literature and although none use curiosity to detect threats and drive the robots' behaviour, they do use powerful techniques which may be applicable to CuWITH. In developing the CuWITH an appropriate method of modelling its curiosity must be determined, there are several such implementations in the literature. These can broadly be divided into curiosity models based on uncertainty, curiosity models based on prediction error and the multi agent cILDA technique.

2.1 Security Robots that have been presented in the Literature

Several security robots have been developed, and are presented in the literature. These robot do not utilise curiosity, however they do address some of the problems that the CuWITH has to face, such as navigation in an indoor environment and threat detection.

2.1.1 iBotGuard: An Internet-based Intelligent Robot Security System using Invariant Face Recognition against Intruder

iBotGuard is a robot security system developed at the Hong Kong Polytechnic University by Liu et al. (2005). Developed on the iBot platform (see Fig 2.1), iBotGuard is intended to use a combination of automated threat detection and robot teleoperation to enhance the capacities of human security officers. iBotGuards intruder detection is built using an invariant face recognition method, implemented in three

Fig. 2.1: iBot robot used for the iBotGuard robot security system.

separate blocks. The first block is responsible for object detection and consists of three agents, a chroma processing agent and a face-region detection agent which together detect regions that contain faces, and a model matching agent which detects faces in these regions. The second block extracts features from faces the first block has detected using a multi-gradient vector flow snakes paradigm. This is implemented using a divide and conquer strategy, with several workers working together to process the data under one master controller. The final block uses graph matching to identify faces. The design of this module assumes that there are several different face databases over a network that iBotGuard has access to. Given this a matching agent is used to move through the network attempting to match a new face to one on the network. A two-step process is used for matching, first a geometric-based coarse matching then a more accurate dynamic-link architecture-based matching. The geometric based matching compares the geometric configuration of the new face with the database face to find a match. The geometric configuration is represented as the position and size of the main facial features (e.g. eyes,nose and mouth), supplemented by the shape of the outline of the face. Matches that pass the geometric based matching are put through the dynamic-link architecture-based matching to evaluate the quality of the match. If the possible match passes the dynamic-link architecture-based matching then it is considered to be a confirmed match and the person is identified.

The iBotGuard system navigates using mobile teleoperation through the internet. The iBotGuard systems robot does not move autonomously, but instead is controlled remotely by a external user such as a security guard. Video from the iBots camera

is streamed to an external controller which processes the video using the invariant face recognition. The results of the computation plus the video are then displayed to the user, who can then move the robot as appropriate to either follow detected threats or search for threats.

In experiments iBotGuard displayed high rates of correct face detection (95%+) with only partial sheltering reducing the detection rate (to 85%). Face recognition rates were also high, with rates of 83%+ despite the face having a pitch or roll of $\pm 20^{O}$. iBotGuards face recognition rate in the face of occlusion and face deformation (the person using different facial expressions and wearing glasses) was lower, but still a relatively high rate of correct recognition was acheived (77% - 82%). The iBotGuard robot was also successfully teleoperated by the team developing it.

2.1.2 *Active People Recognition using Thermal and Grey Images on a Mobile Security Robot*

Treptow et al. (2005) have developed a security robot system which is more independent than iBotGuard and can detect, track and identify people in real time. Their experimental platform is an ActivMedia Peoplebot equipped with a thermal camera and greyscale pan-tilt camera as seen in Fig. 2.2.

Fig. 2.2: The ActivMedia Peoplebot robot used as the experimental platform by Treptow et al. (2005).

The threat detection of the robot consists of three major components.

The first component searches images from the thermal camera for people. If a person is detected the robot will track the person and drive towards them. A

particle filter combined with a simple elliptical model is used to implement the person tracking in this robot, this method was chosen for the speed with which it could be calculated, an important consideration for real-time threat detection. During testing this method was found to correctly detect whether a person was present or not between 81% and 95% of all frames. Furthermore a real time framerate of 80Hz was achieved while limited to the relatively modest computational power of an AthlonXP 1600 processor. This shows not only that this method was accurate, but also that it was fast enough to run in real time while leaving considerable computing resources for other components such as face recognition.

The second component detects faces using the pan-tilt camera. The robot determines the most likely region in the thermal image to contain faces then moves the pan-tilt camera to focus on this area. The face detection algorithm developed by Viola and Jones (2001) is then used to detect any faces in the pan-tilt camera image. This algorithm is well known as being extremely fast and reliable, both necessary characteristics for real-time face detection.

If a face is detected, the third component attempts to recognise it. This recognition is done by first attempting to recognise the face using the well known eigenface method. The result is then used to to update the identity probability (the probability that the unknown person is a particular individual) of the person being tracked using a Bayesian update rule. If over several frames the identity probability goes above a set threshold the face is identified. This component did not perform as well as the first component, with a correct face recognition rate of only 41%. The authors give two reasons for this, the first is that the lighting conditions of the training data were different from the lighting conditions during the experiments, the second is that the recognition rate is dependant on the viewing angle and a very accurately located and cropped face image.

Although the accuracy of the face recognition was not high the robots person detection and face detection were both accurate and fast. All components of the system have been combined in a single system which can run in real time and can perform well in a real environment.

2.1.3 Has something changed here? Autonomous difference detection for security patrol robots

A security robot developed by Andreasson et al. (2007) uses a different approach to detecting threats. Rather than rely on face detection and recognition, Andreassons robot uses the difference between the initial state of its environment and the current state to determine if the security of its environment has been compromised. This state is determined by combining 3D range scans of the environment with SIFT features computed from planar camera images, and storing the data as a Normal Distribution Transform. An additional registration step is used to make the difference calculation robust to changes in time and robot pose. The difference probability of the robots surroundings at a given time t is computed using the spatial and colour difference between the initial environmental state and the environmental state at t. In a real-world experiment in a indoor office the robot was able to detect several environmental changes, including a sliding door that was opened and three chairs and a small box ($\sim 0.4 \times 0.4 \times 0.5 m^3$) that were moved. Furthermore it could detect changes that were invisible to the range scan but could be detected using colour data. These changes included two equally sized boxes that were switched and a coloured strip of paper attached to the floor.

Fig. 2.3: The security robot developed by Andreasson et al. (2007)

2.1.4 Other Security robots that have been Developed

Several other security robots have been developed in addition to these three. The threat detection of these robots tends to be more simple, with several treating the detection of a person in the environment as the detection of a threat (Chien et al., 2005; Luo et al., 2006; Shimosasa et al., 2000). Person detection is often performed using infra-red sensors, sometimes in combination with other sensors such as ultrasonic sensors (Luo et al., 2006) and body sensors (Chien et al., 2005). Not all security robots use threat detection however, the security robot developed by Ryu et al. (2006) is designed to be teleoperated and does not perform any threat detection calculation. Instead video is streamed to the users mobile phone, allowing the user to check the security of the robot environment even if they are a significant distance away.

Many of these security robots also implement some sort of navigation system. Most commonly this is some form of obstacle avoidance where distance sensors are used to detect the presence of obstacles and the robot attempts to navigate around them (Chien et al., 2005; Luo et al., 2006; Shimosasa et al., 2000). Some security robot have additional navigation capacities. The security robot developed by Kim et al. (2006) uses small 'Cricket' nodes (a small, low-power device which broadcasts its position) as points of reference to allow their robot to triangulate its position. This is used to detect and correct errors in the robots path and in combination with a home map table (a map of the robots environment) is used to avoid obstacles and plan a path from the robots current position to the location of some alarm event. The robot then moves along this path to reach the alarm event location. The security robot developed by Shimosasa et al. (2000) patrols an office environment searching for threats. In order to ensure that it keeps on track and does not get lost the robot uses a combination of landmarks and an environment map to help the robot to determine its location. The landmarks are either flat walls that are detected using the robots ultra-sonic sensors or electronic ID cards placed on the floor of the environment in various locations. These cards are read by the robot and matched to the position of the ID card in the environment map.

2.1.5 Motivation of CuWITH Development

Many of the previously discussed security robots possess characteristics that contribute greatly to their utility as a security robot. These characteristics are important for constructing a security robot system and include autonomy, patrol navigation and reliable threat detection.

Autonomy means that the security robot does not require access to external systems to be most effective. This improves the effectiveness of a security robot by making the robot more robust and reliable. Each external system is another failure point for the security system and if that system breaks down it can have serious consequences. For example, if a security robot relies on external RFID devices to determine its location if even one of these systems is damaged it could prevent the robot from properly determining its location. Depending on how the robot handles this situation it could end up stopping or become lost and get trapped against some obstacle. In either case, the robots ability to protect its environment would be compromised and the environment would be vulnerable. An autonomous security robot can avoid this risk. By not relying on external systems the number of failure points is decreased, the robot is made move stable and reliable, much easier to deploy and can handle a greater number of different environments where it may not be possible to install these external systems.

A security robot with patrol navigation can intelligently patrol through an environment searching for threats and can move to investigate any threats that are detected. This is a key advantage of mobile security agents, and without it a security robot becomes little more than an expensive security camera.

Of course the most crucial characteristic of an effective security robot is reliable threat detection. A security robot must be able to reliably detect threats to the safety of the area it is protecting while not wasting any time or resources on perceived threats which are not threats at all. Both of these point are necessary otherwise the security robot will not be able to protect its charge effectively.

Although a few security robot developed previously possess some of these characteristics, none possess all of them. CuWITH is intended to not only posses all

these important characteristics but to also enhance the threat detection capabilities of the robot.

2.2 Curiosity Models and their application to the CuWITH

When considering how to implement curiosity on CuWITH a number of curiosity models were considered. Although a specific method for curiosity-based threat detection has not yet been developed, there are several methods of modelling curiosity which have been presented in the literature.

2.2.1 Uncertainty-based Curiosity Modelling

One method of modelling curiosity is by using a measurement of how certain the system is that its model of the environment is accurate. The more uncertain the system is about the accuracy of a particular part of the model the more curious regions of the environment that correspond to this part are. It is important to note that this method is not based on predictions. This curiosity is computed entirely from its internal model of the environment. Although this model can and often is updated regularly with new environment data the curiosity is not computed directly from this new data.

An example of this method in action can be seen in the DIDO system developed by Scott and Markovitch (1989). The DIDO system represents its environment as a set of classes, with each entity in this environment belonging to one of these classes. In a given environment there is also a set of operations that can be performed on any entity in the environment, and different classes will respond to these operations in different ways. For each class DIDO keeps track of the possible operations that can be performed on it, the possible outcomes of performing these operations and the probability of a given operation on a class having a particular outcome. When the probability of a given outcome drops below a certain limit, the outcome will be removed from the list off possible outcomes for that operation on that class. If a operation on a given class has more than one outcome then its outcome is uncertain. It is this uncertainty that is used to calculate a curiosity value that drives the robots

exploration of its environment. In experiments DIDO was shown to learn the rules of an environment after performing only 6% of the experiments possible in the domain.

Uncertainty-based curiosity is also used by by Uğur, Doğar, Çakmak, and Şahin (2007). Here curiosity is used to drive the robot to learn the traversability affordance of objects for which the traversability is uncertain. The robots environment consists of a flat plane on which a number of obstacles are placed. Four different types of obstacles were used for testing the robot - rectangular boxes, spherical objects and cylindrical objects in either a horizontal or vertical position. The robot uses a 3D laser range scanner to determine the shape, orientation and position of any nearby obstacles. If the robot is already certain of the traversability affordance of a detected obstacle it will be ignored, otherwise the robot will move to interact with the obstacle and obtain information on the affordance of that obstacle. This information is used to train a SVM classifier which represents the robots knowledge of traversability affordance. Whether or not the affordance of an obstacle is certain is determined by a curiosity parameter - as the value of this parameter increases a higher level of certainty is required before the traversability of an obstacle is considered known. If this value is set too high (a value of 1.0 for example) the robot will waste time learning the traversability of uninteresting obstacles (i.e. obstacles that it knows enough about to correctly determine their traversability), if it is too low (i.e. 0.1) the robot will ignore interesting obstacles. The authors selected a curiosity value of 0.5 for their experiments. The robots ability to learn and use its knowledge of traversability to navigate was tested in simulation and in a real environment. In simulation a virtual robot was placed in a simulated environment that contained many different obstacles. By default the robot will attempt to move forward, however if this is not possible it will try other movements. Note that the robot does not simply avoid obstacles but will also drive through obstacles that afford traversability. In the real environment experiment the robot was able to correctly determine the traversability affordance of the box, spherical and cylindrical objects. Furthermore it could also navigate past the obstacles without colliding into intraversable obstacles while still moving through traversable obstacles.

2.2.2 Curiosity Modelling based on Prediction Error

In this method curiosity is based on the prediction error of the system - the difference between what instances the system predicts will be encountered and what instances are actually encountered.

One method of implementing this is to use this error value directly. This error value is then used to drive the system towards interesting instances - instances which are significantly different from what the system expects. This has been implemented in the SAIL robot developed by Huang and Weng (2002). The SAIL robot attempts to learn its environment, using curiosity to focus the learning process. In experiments the robot was allowed to learn its environment then a small toy placed in the environment was constantly moved. This was detected by the SAIL robot, causing it to move to examine the toy and explore this unusual difference. This method is used by several systems (Huang & Weng, 2002; Marshall, Blank, & Meeden, 2004; Barto, Singh, & Chentanez, 2004).

Another method of implementing curiosity modelling based on prediction error first divides the knowledge space of the robot into multiple regions, each of which is learned by its own independent, unique agent. When a prediction of a given situation has to be made the agent of the region which covers this situation is used.

An example of this method in practice is in the Intelligent Adaptive Curiosity (IAC) system developed by Oudeyer, Kaplan, and Hafner (2007). IAC relies on a memory which stores all the experiences encountered by the robot as a set of example vectors. These examples are used to incrementally divide the sensorimotor space into regions, with each region associated with a set of examples. Each region is also assigned an intelligent agent which learns the examples of that region and makes predictions based on what it has learned. When the outcome of a particular situation needs to be predicted, the region associated with that situation is used to make the prediction. The prediction is made, the true outcome is then recorded and the robot computes the prediction error, the difference between what the agent predicted and what actually happened. This error is pushed onto a list unique to the agent. When deciding what situation should be investigated, the robot uses

this list. If the rate of prediction errors for a particular agent is decreasing (i.e. it is progressing rapidly to a strong knowledge of its domain) then situations and actions that that agent covers are more curious. Conversely if the rate of prediction errors is not decreasing (or worse, increasing) the robot will avoid the situations and actions governed by that agent. In both simulation and in a real environment the IAC system demonstrated the ability to focus on situations which it can learn the easiest from, then progressing to more difficult situations. Furthermore it will ignore situations which the system either already knows or is unable to learn. This allows the system to avoid wasting time on known situations and help prevent the robot from getting stuck attempting to learn a situation that is impossible for it to learn.

2.2.3 Curiosity Modelled using Multi-Agent cILDA

The curiosity-driven multi-agent competitive and cooperative linear discriminant analysis algorithm developed by Shimo, Pang, Kasabov, and Yamakawa (2008) is a powerful example of curiosity modelling. This algorithm is based of the Incremental LDA (ILDA) method developed by Pang, Ozawa, and Kasabov (2005). LDA is normally trained in a batch manner, with the LDA system being trained on the training data then used to analyse a set of data. ILDA improves LDA by allowing the trained LDA to be updated, be giving the system the ability to update its knowledge (represented using LDA) with the instances that the system encounters. However, with ILDA all new instances will be used to update the knowledge base, even when these instance are already well known. curiosity-driven ILDA, or cILDA, seeks to overcome this limitation by adding a filter based on curiosity. The essential idea of cILDA is that before using a new instance to update the ILDA the system will first compare this new instance to the existing knowledge base. If the new instance is similar to the instances that are a part of any of the existing knowledge base then it will be rejected because it adds nothing new to the robots knowledge. If the new instance discriminates itself from the classes then it is interesting and it is added to the knowledge of the robot. Thus the curiosity value there is how much

the new instance discriminates itself from the other learned classes - the higher the discrimination, the higher the curiosity. This is then enhanced through the use of multiple agents, with each agent assigned a particular region in the problem space. Each agent then uses cILDA to learn its region. At each timestep thereafter, the agent with the highest curiosity value is activated and allowed to acquire more instances and perform cILDA learning, all other agents are deactivated temporarily.

This method has been tested by observing its ability to handle the Face Membership Authentication (FMA) problem compared to regular LDA. In this problem the objective is to classify a given face as either part of the membership class (i.e. the set of faces the cILDA has been trained on) or the non-membership class. A face database containing 1355 face images of 271 people was used for the experiment, with the membership size ranging from 20 (i.e. 20 membership faces 251 non-membership faces) to 135 (i.e. 135 membership faces 136 non-membership faces) with ten person intervals. Not only was the classification accuracy of cILDA equal to the accuracy of LDA, yet to attain this level of accuracy only required 240 to 420 instances, only 17.7% - 30.9% of the total dataset. cILDA displays fast and accurate classification, and has great potential to be used in a security robot as a method of threat detection. Using cILDA a robot could be easily updated as the office and employees in the office changed over time while still being effective at detecting threats.

2.2.4 Motivation for CuWITH's Curiosity Model

Several different methods of modelling curiosity have been presented here, however these methods are not necessarily applicable to real time threat detection. Many of these models have a particular focus on learning a particular domain quickly and efficiently (Scott & Markovitch, 1989; Uğur et al., 2007; Oudeyer et al., 2007). However, for a security application, these models are not as useful. When a security robot is patrolling an area, it is not trying to learn the environment, rather it is trying to detect threats. These methods may be applicable as a method of quickly learning the environment, however this is not the focus of a security robot.

A curiosity model which is more applicable to a security robot is multi-agent

cILDA. Rather than learning unusual instances, multi-agent cILDA can be modified to register them as threats, and therefore be used as the basis of the CuWITH's threat detection. Furthermore, a threat detection system based on multi-agent cILDA could be trained very efficiently and incrementally - knowledge can be added to the system without retraining it from scratch. However cILDA is a complex algorithm. The computational resources available to the CuWITH are quite limited (see section B for more details) and the execution of cILDA in real time may be beyond its capacities.

A more simple method of modelling curiosity is based on prediction error. There are several such methods (Huang & Weng, 2002; Marshall et al., 2004; Barto et al., 2004; Huang & Weng, 2002) which can be applied to threat detection. For example, the SAIL robot developed by Huang and Weng (2002) learns the environment, then searches the environment for differences, objects which it does not expect in the environment. If such an object is discovered SAIL moves to investigate and learn the curious object. This series of actions requires very little modification to be applied to the problem of threat detection for a security robot, all that is required is replacing the final learning action with some kind of security action. The CuWITH's curiosity is based on this prediction error approach, with the prediction being performed using Principal Component Analysis (PCA). The details of how this is done is explained in Chapter 4.

3. THE DESIGN OF THE CUWITH SYSTEM

When designing the CuWITH system there are a number of properties the system should possess. One of these properties is robustness; if the CuWITH system crashes partway through a security patrol the office is left vulnerable to intrusion. CuWITH must therefore be robust enough to continue patrolling an office for long periods of time without any errors or crashes occurring. CuWITH must also be fast enough to detect threats in real time. If not, then CuWITH will be unable to react fast enough to check the office in real time, allowing intruders to easily evade it. CuWITH must also be easy to maintain and modify if it is to be an effective platform for further research. Designing the CuWITH to be easy to maintain and modify encourages exploration of the CuWITH's capabilities, if a researcher want to test a particular curiosity model this can easily be integrated into the CuWITH and tested. If CuWITH is difficult to maintain and modify it will discourage changes to the CuWITH and therefore stifle research the relies on the CuWITH. To ensure the CuWITH displays these properties a modular design strategy is employed.

3.1 Design Strategy

The key architectural strategy employed when developing the CuWITH system was the use of a modular design strategy. This strategy divides the system into components that can be developed and used independently from one another. There are several reasons why this strategy improves the quality of the CuWITH system

Use of a modular design strategy allows much more efficient and powerful testing of the system. Individual components can be separated from the rest of the system and tested individually to ensure each component is performing its assigned task reliably. If an issue is discovered during this individual testing, the cause can be

rapidly narrowed down. This is because it can be guaranteed that the cause is within the component being tested, not with any other component. If the design is not modular this individual testing is not possible. if a problem is detected, the cause could be within several components and narrowing the cause is much more difficult. Even if one component requires another ancillary component to work, through the use of constant interfaces a fake ancillary component can be supplied to the component being tested. This again allows this specific component to be tested without the risk of bugs in other components contaminating the testing procedure. For example, when developing the programmable navigation component obviously it needs to control the robot (through the robot control component). However testing the programmable navigation component using the robot is time consuming and the source of any bugs is unclear (is it a issue with the robot interface or the navigation instructions?). To resolve this issue a fake robot controller was developed that did no communication with the robot, it only printed out all movement instructions it was asked to perform. This was immensely helpful for testing because it was immediately clear not only if there were any issues with the programmable navigation component but also a good idea of what the issue was (e.g. a velocity value was set to high). By improving the testing that can be performed on the system, modularity can improve the reliability and stability of code, both necessary qualities of a security system.

Maintainability is also greatly improved by using a modular design strategy. This is best demonstrated by considering maintenance of a system which is not modular i.e. one which each component is tightly coupled to other components. If a system is not modular then any change to a single component in the system could propagate through the entire system, causing other components to fail. This happens because other components depend on the modified component - they may for example expect the component to contain a particular function component. If this signature is modified or removed components that are dependant on it will fail. Fixing such a situation would not only require changing the modified component, but also every other component dependent on it. Furthermore these changes could result in needing to change other components that are dependent on the dependent

components. This can quickly result in the entire system needing to be modified to accommodate a change in a single component. This makes any modification potentially difficult and time consuming. These problems can be avoided by using a modular design. Since each component is not dependent on other components, changes to one components will not propagate and are relatively straightforward.

Modular design also allows components of the system to be easily reused. If the architecture of the CuWITH system is significantly modified many of the components of the current system can be reused in the new system. This is because of their modularity - the components are not dependent on the architecture of the system or other components. Because of this components can easily be moved between systems without losing any functionality or requiring additional extraneous code to be added to the system. This greatly speeds the development of the system, rather than needing to 'reinvent the wheel' by writing and debugging a new version of a component the current version of the component can be used. This component has already been tested to ensure stability and reliability and thus does not need intensive development. This saves time and effort to focus on other components of the new system, such as the components which are being significantly changed.

Despite the advantages of the modular design strategy in the current CuWITH system there are still some dependencies. This is because CuWITH must not only be robust and maintainable, it must also be fast. To improve the speed of the CuWITH system therefore certain components were made more tightly coupled. Also, several components of the CuWITH are responsible for initialisation and communication between components and, as a result,are more dependant on other components.

3.2 Design of the CuWITH System

Based on this strategy a design for the CuWITH system has been developed, and can be seen in Fig. 3.1. As can be seen there are many components within this design, covering such tasks as threat detection, patrol navigation and curiosity driven movement. These components can be divided into three groups, threat detection and tracking, robot navigation and shared global data. The components of the

threat detection and tracking group can be seen on the left side of Fig. 3.1. These components are the focus of this research and include the crucial curiosity based threat detection and curiosity driven threat tracking components. On the right of Fig. 3.1 is the robot navigation components. These components are executed in a separate thread from the threat detection and tracking group, this is due to the manner in which the CuWITH robot is controlled. Moving the CuWITH consists of three steps; instructing the CuWITH to match a particular velocity, waiting until the CuWITH has moved the desired distance, then instructing the CuWITH to halt. Because of the second step in this process the thread is executing the navigation components will stop and wait while the CuWITH moves. To prevent the robot navigation from bringing the CuWITH system to a halt while the robot moves the navigation is executed on a separate thread from the other components of the CuWITH. However the robot navigation components and the threat detection and tracking components still need to communicate, to that end a global thread interface is provided which allows the two threads to communicate in a safe and reliable manner. These components are discussed in more detail below.

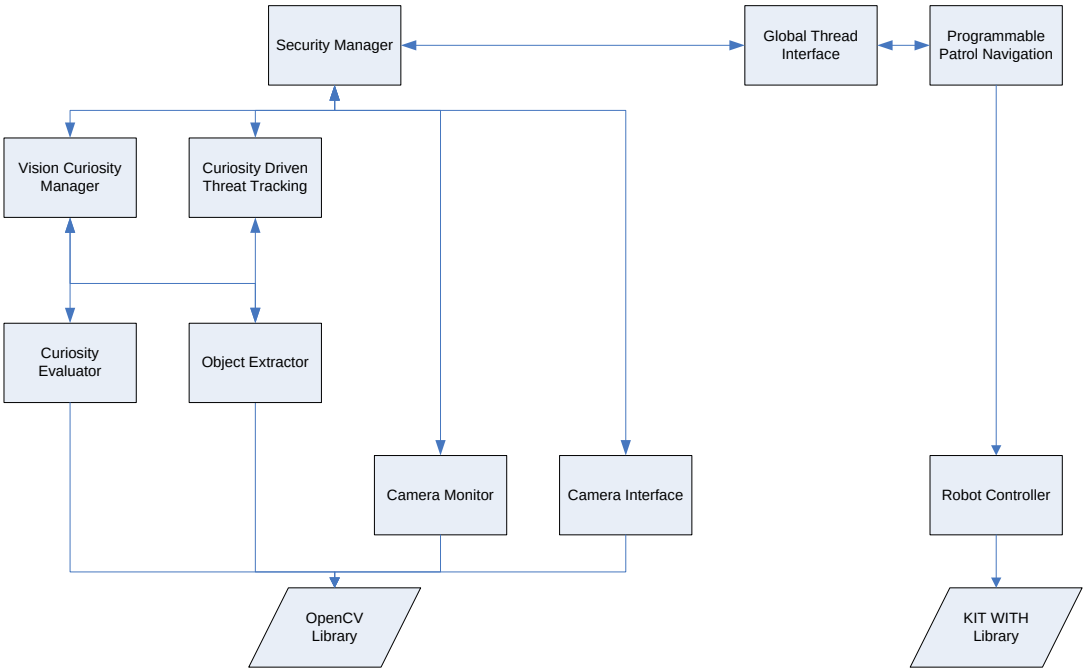


Fig. 3.1: Structure of the CuWITH system

3.3 Core Components of the CuWITH

The CuWITH's has two core components that are the focus of this research; the curiosity based threat detection and curiosity driven threat tracking components.

3.3.1 Curiosity Based Threat Detection

The curiosity based threat detection component is responsible for modelling curiosity and for determining if a threat is present in the environment. It is the most computationally demanding component of the CuWITH, as it must process a large amount of data (an image can consist of upwards of 19200 pixels) in real time. As such, when designing the curiosity based threat detection component dependency minimisation must be balanced against real time execution. With this in mind the curiosity based threat detection component can be split into two subcomponents; the vision curiosity manager and the Curiosity Evaluator.

The vision curiosity manager has three key responsibilities. Firstly it must obtain images of any objects in the CuWITH's line of sight. This is implemented by using the camera interface to get an image from the camera, then using the object extractor to extract from the camera image a sub image containing an object (an object image). Secondly the vision curiosity manager must determine whether or not a given object image is curious, i.e. if an object in the image is not an object CuWITH is expected to encounter in the environment. This classification is performed using the curiosity value of the object image (computed using the Curiosity Evaluator) and is explained in section 4.2. However, the most important responsibility of the vision curiosity manager is to make a final decision on whether or not there is a threat in the environment. This decision is made using Eq. 4.8 in section 4.2 and is based on the ratio of curious to non-curious faces detected in the CuWITH's recent history. This responsibility is the most important of the three and the basis for the security capabilities of the CuWITH. As a result of these responsibilities the vision curiosity manager is dependant on the camera interface, object extractor and Curiosity Evaluator. This dependency has however been limited by hiding the details of how the camera interface, object extractor and Curiosity Evaluator work

from the vision curiosity manager. The vision curiosity manager does not need to know about or rely on how exactly the Curiosity Evaluator works or what settings it has, for example, and as a result the Curiosity Evaluator can be modified without affecting the vision curiosity manager.

The Curiosity Evaluator is trained on the set objects that are expected in the environment, and is responsible for computing the curiosity value of a single given object image. The curiosity value is a measure of how different the encountered object is from the CuWITH's expectations, the computation of the curiosity value is explained in detail in sections 4.1 and 4.2. The knowledge storage and curiosity computation is completely encapsulated within this subcomponent, this allows overhead to be reduced and improves the efficiency of this subcomponent. Furthermore, this subcomponent is only dependant of the OpenCV library used to implement the curiosity calculation, protecting it from changes to the CuWITH system.

The detection of threats is the most important responsibility of the curiosity based threat detection component, however when a threat is detected some action must be taken.

3.3.2 Curiosity Driven Threat Tracking

CuWITH must not only detect threats using curiosity modelling, it must also be driven by curiosity to track these threats so that they may be neutralised. This role is performed by the curiosity driven threat tracking component. The responsibility of the curiosity driven threat tracking component is simple; to compute a robot movement that will bring the threat into the center of the CuWITH's field of vision and a set distance in front of the CuWITH. This computation is described in more detail in section 4.3. This component is dependant on the physical details of the CuWITH robot, as the tracking movements are computed assuming a set velocity of the robot. It is also dependant on the resolution of the camera image. However, the curiosity driven threat tracking component is not dependant on any other component of the CuWITH. This (as with the Curiosity Evaluator) protects the curiosity driven threat tracking component from changes to the CuWITH system, making

the component more robust and maintainable.

3.4 Ancillary Components of the CuWITH

The curiosity based threat detection and curiosity driven threat tracking components are the most important components of the CuWITH for this research, however these components cannot be used alone. For CuWITH to be effective as a security robot there are a number of components that, while not directly related to curiosity-based security, are nonetheless crucial. Of particular note are the programmable patrol navigation, the object extraction component, the global thread interface and the security manager.

3.4.1 Programmable Patrol Navigation

To be effective as a mobile security agent the CuWITH must be able to move in its environment and actively search for threats. If the CuWITH cannot move in the environment, then it is no better than a stationary security camera. One of the strengths of a mobile security agent is that it is mobile and can patrol the environment actively searching for threats, the CuWITH must be able to execute such a security patrol if it is to be effective in this role. It is towards this end that the programmable patrol navigation component was developed. This component computes the instructions necessary for the CuWITH to move from one point in a given patrol path to another and sends the instructions to the robot. When a security point is reached (i.e. a point that needs to be checked for the presence of threats) the navigation thread halts the robot and waits for the presence of a threat to be either confirmed or denied. If it is denied the programmable patrol navigation component will continue moving the robot through its patrol. Alternatively, the programmable patrol navigation component will wait for and execute and driven movement instructions sent to it from the security manager. This component has been enhanced by making it programmable. The patrol path is not hard coded into the component, rather a script is read in by the programmable patrol navigation component at runtime. This script defines the patrol path that the CuWITH will

execute. This improves the flexibility and maintainability of the CuWITH as if the robot is moved to a different environment or the current environment is changed the system does not need to be modified at all, only the patrol path script need be changed. Furthermore, the programmable patrol navigation component is only dependent on the robot interface and the global thread interface. This protects the programmable patrol navigation component from any modifications to the threat detection or threat tracking of the CuWITH, communication between these components is performed through the global thread interface.

3.4.2 Object Extraction

The responsibility of the object extraction component is to search a given camera image for particular types of object and extract from the camera image the sub-image containing the object. This is crucial as the threat detection of the CuWITH is computed based on object images - without this component to supply the CuWITH with these images threat detection is not possible. Furthermore, this component must be accurate and have low computational demands. Inaccurate object extraction can deteriorate the accuracy of the threat detection and lead to the detection of non-existent threats. High computational requirements could cripple the CuWITH and render it unable to search for threats in real time, also the more processing power required by the object extractor, the less processing power there is available to other components such as the curiosity evaluator. The techniques applied to this task are discussed in Sec. 5.2 Apart from the OpenCV library this component has no dependencies.

3.4.3 Security Manager

The security manager controls the threat detection and threat tracking of the CuWITH system. It is the responsibility of the security manager to determine when a security check is required, initiate the security check and decide on an appropriate course of action based on the results of the security check. The security manager decides when to perform a security check using the current location of the CuWITH. The patrol path will have certain points in the patrol classified as security points, if

the CuWITH has reached these points then a security check needs to be performed. If a security point has been reached, the security manager instructs the curiosity based threat detection component to perform the check, then examines the results. If a threat has been detected it is tracked by determining the location and size (in pixels) of the threat in the camera image, then supplying this data to the curiosity driven threat tracking component to determine an appropriate movement to follow the threat. This movement is then sent to the programmable patrol navigation component to be executed by the robot. Because the security manager is intended to control several components and interface with the programmable patrol navigation component, it is dependant on several other components. This is a necessary price to pay; to completely eliminate this dependency would be extremely difficult and the resulting code would be difficult to maintain and would be slower.

3.4.4 Global Thread Interface

The CuWITH system has two threads running co-currently, the main thread which handles threat detection and tracking and the navigation thread which is concerned with the movement of the robot. If these two threads cannot communicate and synchronise with each other, the CuWITH will not be able to function - the security tests may not be performed at the correct locations (or at all) and driven instructions may be ignored by the navigation thread. To prevent this from happening a robust inter-thread communication module is necessary. The global thread interface fulfils this role. Communication between threads is achieved through shared data and mutexes encapsulated by public functions of the global thread interface. This ensures that all communication between the two threads is thread safe while still maintaining the speed necessary for real time security.

3.4.5 Other Components

Along with the above major components there are three tertiary components that are focused on hardware access. The robot controller is an example of this. The robot controller acts as the interface between the WITH robot and the CuWITH system. Essentially it is a C++ wrapper around the WITH control library supplied

to KEDRI by KIT, containing functions for setting the robots velocity and accessing its distance sensors (among other things). To limit dependency on this component all other components that depend on the robot controller actually depends on a robot interface. This is a static interface which the robot controller implements, however because other components are dependant on the interface rather than the implementation changes to the robot controller do not affect any other components. The interface also allows the real robot controller to be easily replaced with a fake controller that merely prints out the instructions it is given. This is extremely useful for testing and the change required is a simple case of removing one line of code and replacing it with another.

The camera interface is a similar example, however this component controls access to the CuWITH's camera. It is also responsible for ensuring the camera image has the resolution, colour depth and colour model that the rest of the system expects. As with the robot controller, rather than being dependant on the The Camera controller other components are dependant on a camera interface which is implemented by the Camera controller.

The third tertiary component is the camera monitor. During the development of the CuWITH the ability to view what the robot was seeing became desirable. This was for reasons of testing and display. When fine-tuning and debugging the curiosity-driven threat detection it was very helpful to be able to see what the robot is seeing, along with any faces detected and whether they were curious or not. This is because although it is possible to record when and where a face is detected for example, it is more difficult to record the contextual information. The easiest way to view and record this information is to display the camera image, along with some relevant information (e.g. highlighting faces that have been detected). This way any issues with the object detection or curiosity-driven threat detection can be identified and resolved more quickly. It is also useful for display purposes. When demonstrating the CuWITH being able to display the camera image along with some information about the threat detection process (i.e. highlighting faces that are curious and not curious) is very helpful to explain what the robot is doing. It

also has a real security application, using this component allows security officers to monitor what the robot is seeing. This component was implemented with two modes. One mode would simply display an image on-screen, this was more useful when unit testing the curiosity calculation. The other mode transmitted the image over wireless link to a server on a desktop which would then display the image on the desktops screen. This was useful when displaying the robot and when performing full system testing on the CuWITH (the curiosity process is made clearer).

3.5 CuWITH Process Control

3.5.1 Execution Process

CuWITH's execution procedure is split into two threads, with the Programmable Navigation component and Robot Control component associated with the navigation thread and the remaining components (including the Threat Detection Manager) associated with the main thread. The execution procedure of these threads during a security patrol can be seen in Fig. 3.2, with both threads going through several stages. Initially both threads start in the navigation stage. In this stage CuWITH moves along its patrol path until it reaches a security point. During this stage the main thread only updates the camera monitor, no threat detection is performed. Instead the navigation thread moves the CuWITH robot along the patrol path and at each step updates the current location of the robot. Both threads check if the robot has reached a security point at every iteration - if a security point has been reached, the threads transfer to the Security Check state. It is in this state that threats to the safety of the environment are searched for and detected by the main thread. During this process the navigation thread stops the robot and waits for the main thread to decide if there is a threat present in the environment. If a threat is not detected both threads return to the patrol navigation stage and continue the patrol. However if a threat is detected both threads switch to the Curiosity Driven stage, where the CuWITH is driven by its curiosity of the threat to follow the threat. This is done in two sections, first the main thread computes a driven move instruction set which will move CuWITH to get a better view of the threat.

This instruction set is then sent to the navigation thread which executes this move, then waits for further driven move instructions. This process will continue until a set timeout, at which point the main thread will switch to the Continue Patrol stage and send a last 'driven move' to move the CuWITH back to the security point. The main thread will then set the current security point as being checked and both threads will then transfer back to the Patrol Navigation stage, starting the process again.

3.5.2 Thread Synchronization

Key to this design is the communication and synchronisation of the CuWITH systems two threads. This is done through the Global Data Storage (represented in Fig. 3.2 as the area between the two threads). This is implemented as a single data structure that encapsulates everything necessary for communication and synchronisation between the two threads.

The key communication that occurs between the two threads is the transfer of the driven move instructions from the main thread to the navigation thread. Here the global data storage is used as a middleman between the two threads, with the instructions being first sent to the global data storage. These instructions are then read by the navigation thread and executed. Synchronisation is implemented by using the global variable to store the state of the CuWITH. In particular, whether or not the robot has reached a security point is determined by the Global Data Storage. Both threads check the Global Data Storage to determine if the robot has reached a security point, and if so transfer to the Security Check stage. In this way it is ensured the threads will not be in conflicting states. This prevents errors and system instability that can be caused if the two threads start working at cross purposes.

One of the most important elements is the use of mutexes to control access to shared data, especially if one thread modifies the data. If mutexes are not used, the system runs the risk of one thread accessing data partway through that data being modified by the other thread. This results in corrupted data being read by

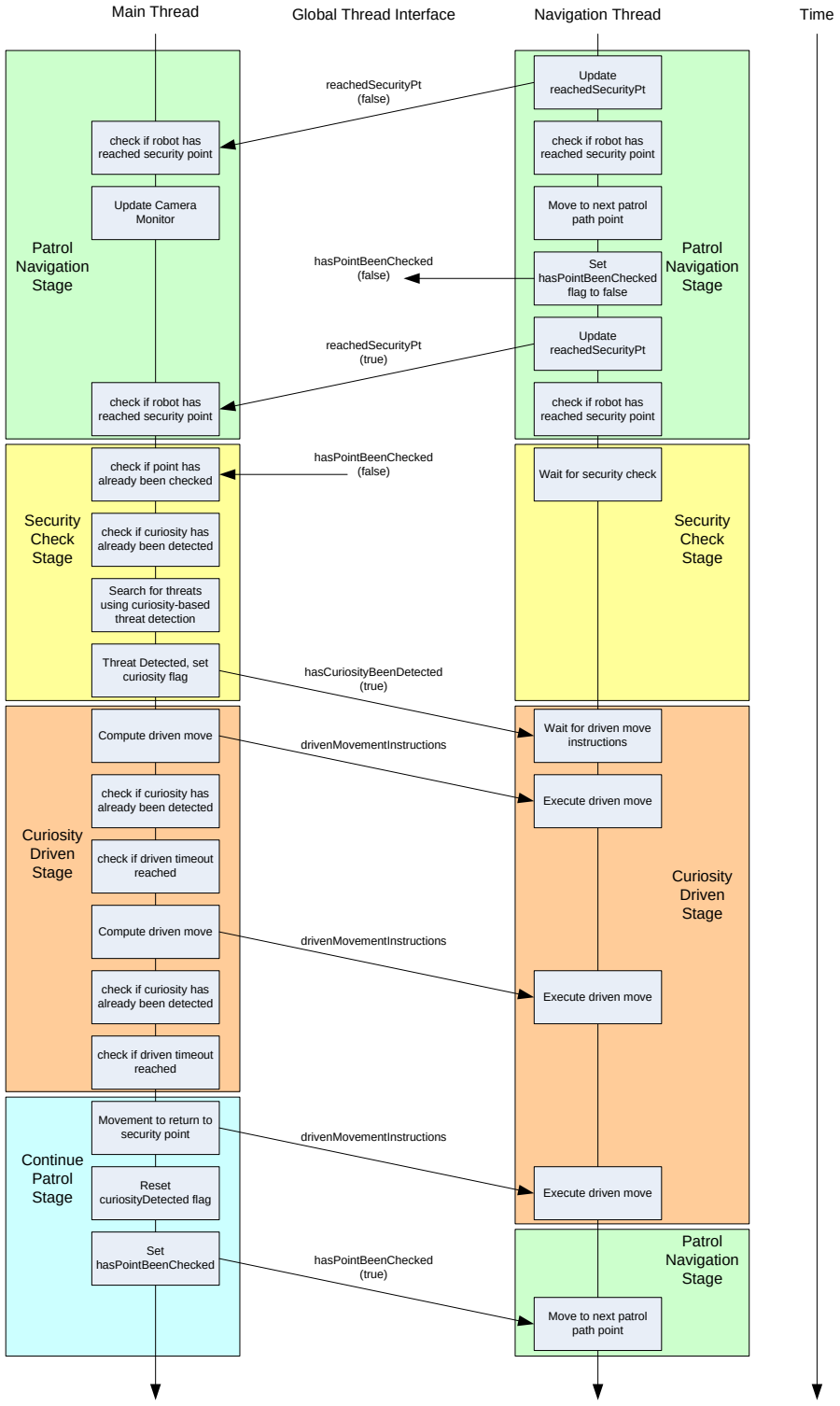


Fig. 3.2: Process diagram of the CuWITH system

the thread. This can easily cause the system to fail. However, mutexes are easy to forget to use, and even one data access not protected by a mutex (or worse, a data access that locks the mutex without unlocking it afterwards) can cause system failure. This is solved in CuWITH by accessing shared data through public functions

of the Global Data Storage. These functions internally ensure that the shared data is properly protected by mutexes. This protects the system from human error and improves the systems stability.

4. CURIOSITY BASED THREAT DETECTION AND CURIOSITY DRIVEN THREAT TRACKING

The core of the CuWITH security system is curiosity based threat detection and curiosity driven threat tracking - a security robot must be able to both detect threats to the safety of its environment and track any threats detected. In the CuWITH this has been implemented through the use of curiosity modelling to detect threats, and curiosity-driven threat tracking to follow any detected threats. The curiosity principal applied here is that CuWITH can expect to encounter a particular set of objects in the environment - people who are allowed in the environment, for example. If a threat intrudes in this environment by definition it will not be within this set of expected objects and thus will be curious, something outside the norm. It is this difference which is harnessed for threat detection in the CuWITH. Although these techniques are general enough to be effective against any threatening object, the current implementation on the CuWITH is focused on the face curiosity case study. For the face curiosity case study the threat detection component learns the faces of a set of known people, then compares any encountered faces with this set of known people to determine if the encountered person is known, or a threat.

4.1 Feature Extraction for Curiosity Modelling

For the curiosity modelling to be effective it requires a set of discriminating features to differentiate between objects that the CuWITH expects to encounter in its environment and objects that are completely new to the robot. In CuWITH these features are extracted using PCA, a feature extraction technique that is well known in the literature (Treptow et al., 2005; Turk & Pentland, 1991; Barreto, Menezes, & Dias, 2004). The first step in this process is to train the feature extraction. Assume

$X = \{\vec{x_1}; \vec{x_2}; \vec{x_3}; \dots \vec{x_N}\}$ represents the set of normalised image vectors used to train the robot (where each image vector is a vectorised image of an object). Given this, the average vector $\vec{\mu}$ of X is defined as

$$\vec{\mu} = \frac{1}{N} \sum_{i=1}^{N} \vec{x_i}. \tag{4.1}$$

Each image vector differs from $\vec{\mu}$ by vector Θ

$$\Theta_i = \vec{x_i} - \vec{\mu} \text{ for } i = 1 \dots N. \tag{4.2}$$

The discriminate feature space is represented using an eigenvector matrix $U = \{\vec{u_1}; \vec{u_2}; \dots \vec{u_M}\}$, obtained by solving the following eigen-decomposition function:

$$UC = \lambda U^T, \tag{4.3}$$

where λ is the eigenvalue matrix and C is the covariance matrix of X, computed using the following formula:

$$C = \frac{1}{N} \sum_{i=1}^{N} \Theta_i \Theta_i^T. \tag{4.4}$$

Although this will represent the discriminant feature space not all of the eigenvectors in U are required to represent it fully. U' is the subset of U containing the M' ($\ll M$) eigenvectors with the largest associated eigenvalue. The features for each image of an object are found by the security robot using the following formula:

$$y = U'^T(\vec{x} - \vec{\mu}), \tag{4.5}$$

where $\vec{x}$ is the image vector and y is the features of $\vec{x}$.

4.2 Curiosity Modelling for Threat Detection

The purpose of curiosity modelling is to model the robots expectations of its environment and to determine if an object is outside these expectations. If such a curious object is detected, it is classified as a threat.

Assume there is a trained dataset X_{tr} that represents the knowledge the CuWITH has about the objects it can expect to encounter in its environment. This dataset consists of the features (computed using Eq. 4.5) of each training image vector $\vec{x_1} \ldots \vec{x_N}$. X_{tr} is split into clusters $X_{s1}, X_{s2} \ldots X_{sm}$, with each cluster representing a different part of the robots knowledge. In practice, the mean feature vector of each cluster is used to represent each part of knowledge. As a new object image $\vec{x}$ is acquired by robot, the discriminating features y are extracted from it using Eq. 4.5. Thus, to evaluate the difference of $\vec{x}$ to cluster $X_{si}, 1 \leq i \leq m$, a mathematical residue is defined

$$\xi_i = \|y - \bar{x}_{si}\| \tag{4.6}$$

where $\bar{x}_{si} = \frac{1}{|X_{si}|} \sum_{\vec{x_j} \in X_{si}} \vec{x_j}$ is the mean vector of cluster X_{si}.

Thus, the general difference over the whole office environment is computed as,

$$\langle e \rangle = \min(\xi_1 \ldots \xi_m) \tag{4.7}$$

The above residue is applied to detect threats. If the calculated $\langle e \rangle$ is greater than a pre-specified curiosity threshold ω, then the robot has encountered a curious object. A curious object is not supposed to be present in the CuWITH's environment and thus constitutes a security threat.

However, this more simple curiosity calculation is unusable in a real environment because of the inevitable noise that the robot will encounter. Differing lighting conditions and shadows can cause the curiosity value of an object to fluctuate, decreasing the stability of the curiosity value. Furthermore, rarely the CuWITH's object extraction will detect a false positive - for example, a blank wall that was been lit to look like a face. These false positives will result in a high curiosity value (since there isn't even an object present, let alone a known object) and may trigger a security response for a threat that is non-existent. To overcome these issues the curiosity values over a set period of time κ are considered. Given that over κ there were $\langle e_1 \rangle \ldots \langle e_p \rangle$ curiosity values computed, a combined curiosity value $\mathcal{C}$ is

calculated as follows:

$$\mathcal{C} = \frac{\sum_{i=1}^{p} c_i}{p}, \tag{4.8}$$

where c_i is 1 if the ith object detected is curious, and 0 otherwise. $\mathcal{C}$ is therefore the percentage of objects detected over time period κ that are determined to be curious by CuWITH. If $\mathcal{C}$ is greater than a threat threshold ϵ, then a threat has infiltrated the environment. The CuWITH's next course of action is to track the threatening object as described in the next section.

4.3 Curiosity Driven Threat Tracking

When developing the CuWITH some form of security action was desired. Raising an alarm was a trivial action to implement, so a threat tracking component was developed to allow the CuWITH to follow detected threats in the environment. This threat action could intimidate a intruder and cause them to leave, it could also be used by security forces to locate the threat and neutralise it. For driven movement both the robot and the threat are considered to be in a 3D environment, with a j, k and l axis as seen in Fig. 4.1,

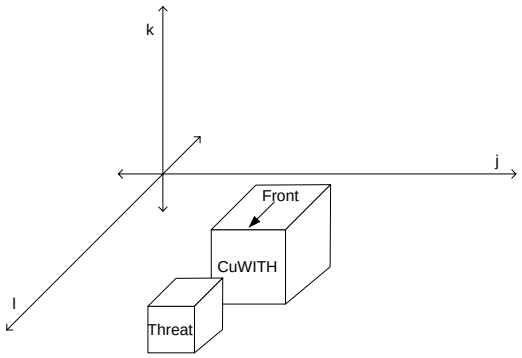


Fig. 4.1: The 3D representation of the office environment space used for threat tracking

The direction of these axes is relative to the robot, with the j-axis extending to the CuWITH's left and right, the k-axis extending above and below and the l-axis extending in front of and behind the CuWITH. Given this there are two axis the robot can move along, the j-axis (left and right) and the l-axis (forward and backward). The vertical axis is ignored for driven movement for the simple reason

that the CuWITH can neither tilt its camera or move up and down. Thus even if the threat does move up or down there is no action CuWITH can take to follow it. Movement along the j-axis keeps the threat within the CuWITH's field of view, while movement along the l-axis can keep the robot within a certain distance of the threat. Both movements are defined using two values; a direction r which has a value of either left,right,forward or backward, and a wait time T_{wait} which determines how far the CuWITH should move in that direction (the CuWITH's velocity in a particular direction is a set constant).

The j-axis driven move is determined using the difference j_{diff} between the centre of the camera image w_{mid} and the centre of the threat w_{obj} (see Fig. 4.2) as a measure of how far to the left or the right of CuWITH the threat is, computed using the equation $j_{diff} = w_{obj} - w_{mid}$.

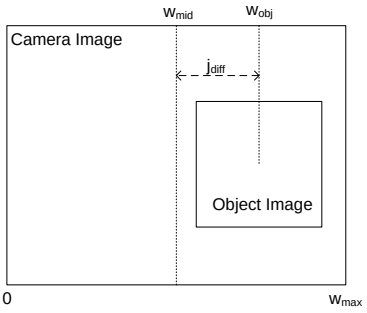


Fig. 4.2: The computation of the objects horizontal displacement j_{diff} using the center of the camera image w_{mid} and the center of the object w_{obj}

With j_{diff}, a j-axis movement can be calculated as follows:

$$T_{wait} = |j_{diff}|.S_j.T_{const}, \tag{4.9}$$

$$r = \begin{cases} \text{left} & \text{if } j_{diff} < 0 \\ \text{right} & \text{else} \end{cases} \tag{4.10}$$

The S_j term in Eq. 4.9 is a sensitivity term that determines the degree to which CuWITH reacts to a given value of j_{diff}; higher values of S_j will result in the robot moving further for a given value of j_{diff}, while conversely lower values of S_j will have the opposite effect. The result of $|j_{diff}|.S_j$ is shifted to a time in milliseconds by multiplying the result with a constant T_{const} (set to a value of 15). The direction

of the calculated driven move is computed using the sign of j_{diff} as seen in Eq. 4.10 - a negative value results in a move to the left, a positive value results in a move to the right.

The l-axis movement of the robot is computed in a more complex manner. It is based on the principle that as an object approaches the camera, the area (number of pixels) it takes up on the camera image increases; conversely, as it retreats from the camera the area it takes up on the camera image decreases (see Fig. 4.3).

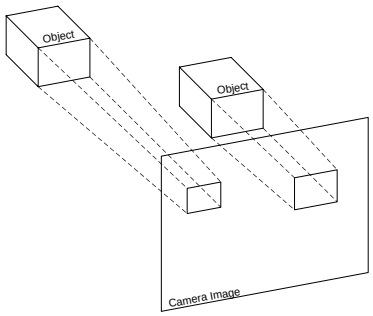


Fig. 4.3: This illustrates how the distance between the robot and an object is determined - assuming the size of the object is constant, its apparent size is proportional to its distance from the robot.

Based on this principle how far away a target is from the camera can be estimated using the area it takes up on the camera image. This is based on the assumption that the size of the object is constant and in some sense known. In the face-curiosity use case implemented in CuWITH, it is assumed that all faces are approximately the same (real) size and thus if the face is the desired distance from the robot it will always take up a particular area A_{opt} of the camera image. The distance the robot needs to move to maintain the desired distance from the threat can therefore be determined, based on the difference between A_{opt} and the actual area the face takes up in the camera image A_{obj}:

$$l_{diff} = \sqrt{A_{obj}} - \sqrt{A_{opt}}, \tag{4.11}$$

where l_{diff} is the estimated distance the object is from the robot. The l-axis move-

ment T_{wait}, r is thus defined as follows:

$$T_{wait} = |l_{diff}|.S_l.T_{const}, \tag{4.12}$$

$$r = \begin{cases} \text{forward} & \text{if } l_{diff} < 0 \\ \text{backward} & \text{else} \end{cases} \tag{4.13}$$

where S_l is the sensitivity variable for l_{diff}. The direction of the calculated driven move in this case is computed using the sign of l_{diff} as seen in Eq. 4.13 – a negative value results in a move forward while a positive value results in a move backward.

Both calculated moves have to be larger than a set threshold Γ_{min} otherwise they will be ignored. This is for two reasons. Firstly very small moves of the CuWITH are insignificant with regards to keeping the object in a particular position relative to the robot. The second reason is because very small movement instructions – particularly many small move instructions - can cause the robot to turn slightly to one side, resulting in the robot deviating from its path. This threshold has the useful side effect of preventing another issue. Data on the position and area of the face in the camera image is prone to fluctuations due to the realities of a real environment. Even subtle changes in lighting - changes too small to be easily detectable by the human eye - can cause the position or area of the face to change slightly even though the person may not be moving. Putting a minimum limit on the size of the movement prevents the CuWITH from constantly making small adjustments in position to match non-existent changes in the person's position.

A second threshold Γ_{max} is used to limit the distance the CuWITH robot moves in a single driven move. If $T_{wait} > \Gamma_{max}$ then $T_{wait} = \Gamma_{max}$. This is done to facilitate real-time driven movement. If CuWITH spends too much time executing a driven move it can miss the current motion of the threat. This can cause the robot to move away from the threat because it is unable to keep up with the speed with which a threat can change direction.

After the application of these thresholds there are three possible outcomes. The first is that no move passes the thresholds, in which case no movement is performed by the robot. The second possibility is that only one of the possible movements

(either left/right or forward/backward) passes the thresholds, in which case that movement is executed by the robot. The third possibility is that both possible movements pass the thresholds, in this case only the largest movement is performed, the smaller movement is ignored.

4.4 Implementation of Threat Detection and Tracking

The previous sections introduce the techniques we used for curiosity-based threat detection and curiosity-driven threat tracking. In CuWITH these techniques are implemented as two separate algorithms, one for detecting threats (Alg. 1), the second for computing an appropriate driven move for the robot (Alg. 2).

For threat detection (see Alg. 1) the CuWITH system is supplied with the trained dataset X_{tr} as described in section 4.2. This dataset is trained separately from the CuWITH system but all of the training images are taken using the robots camera. When a security check is required, the curiosity-based threat detection algorithm is executed and a boolean value is returned to the system - either 'true' if a threat is detected or 'false' if no threat is detected. Note that there is an additional test to ensure that p is larger than a threshold p_{min}. This is to ensure the stability and reliability of the threat detection. Although the object detection of the CuWITH is fast and reliable, it is not perfect. Rarely it will detect an 'object' which does not actually exist. p_{min} prevents these errors from triggering a response from the CuWITH.

Assuming a threat has been detected the robot must track the threat as described in section 4.3. This is implemented as in Alg. 2, with the algorithm receiving the position j_{obj} and area A_{obj} of a object image and returning a appropriate driven move to the system.

Note that there is an additional test on line 3, this is to catch early situations that will result in no driven movement. Here j_{min} and l_{min} are the smallest values j_{diff} and l_{diff} respectively can have before becoming insignificant. If j_{min} and l_{min} are both insignificant then T_{wait} is set to 0. If the returned driven move has $T_{wait} = 0$ (as is the case at lines 4 and 14) then no driven move will be performed. When

Algorithm 1 Curiosity-Based Threat Detection Algorithm

Require: Matrix X_{si} representing the CuWITH's knowledge and the minimum number of objects that must be processed for the curiosity to be valid p_{min}
Ensure: curiosity is either **true** or **false**

1: initialise number of object images processed p to 0
2: initialise number of curious objects detected q to 0
3: **repeat**
4: Obtain object image
5: Normalise object image to a 60x60 image x with an 8-bit RGB colour space
6: Extract red colour channel vector $\vec{x}$ from normalised object image x
7: Extract features y from $\vec{x}$ using Eq. 4.5
8: **for** X_{si} for $i = 1 \ldots m$ **do**
9: compute difference between y and X_{si} using Eq. 4.6
10: **end for**
11: select smallest distance value $\langle e \rangle$ across all clusters
12: **if** $\langle e \rangle < \omega$ **then**
13: increment q
14: **end if**
15: increment p
16: **until** $t > \kappa$
17: $\mathcal{C} = \frac{q}{p}$
18: **if** $p < p_{min}$ **then**
19: **return false**
20: **else if** $\mathcal{C} < \omega$ **then**
21: **return false**
22: **else**
23: **return true**
24: **end if**

determining along which axis the robot should move (the j-axis or the l-axis) at line 6, rather than a simple comparison the difference between ($|j_{diff}|$ and $|l_{diff}|$ is compared to a threshold ϕ. This is used to compensate for the inherent variability of $|j_{diff}|$ and $|l_{diff}|$; both values can vary even if the threat is not actually moving relative to the robot.

4.5 Summary

To surmise, there are two key components of the CuWITH system with regards to security; curiosity based threat detection and curiosity driven threat tracking. Threat detection is obviously necessary for a security robot, and is implemented using a curiosity based paradigm discussed in section 4.3. Although based on the well-known eigenface face recognition method developed by M. Turk et all. (Turk &

Algorithm 2 Curiosity Driven Move Algorithm

Require: $j_{obj} >= 0$, $A_{obj} >= 0$, $j_{mid} >= 0$, $A_{opt} > 0$
Ensure: $T_{wait} >= 0$, r is one of LEFT,RIGHT,FORWARD,BACKWARD

```
j_diff = w_opt − w_obj
compute and A_diff as seen in Eq. 4.11
if j_diff < j_min and l_diff < l_min then
    return T_wait = 0
else
    if (|j_diff| − |l_diff|) > φ then
        compute T_wait using Eq. 4.9
        determine r (LEFT or RIGHT) using Eq. 4.10
    else
        compute T_wait using Eq. 4.12
        determine r (FORWARD or BACKWARD) using Eq. 4.13
    end if
    if T_wait < Γ_min then
        T_wait = 0
    end if
    if T_wait > Γ_max then
        T_wait = Γ_max
    end if
    return T_wait, r
end if
```

Pentland, 1991), curiosity-based threat detection expands the eigenface method by not attempting to specifically identify specific people in its environment. Rather, all that is required is a single classification; does the detected face belong to one of the people allowed in the environment or not. Because of this focus on distinguishing threat from non-threat rather than individuals, the threat detection system is not as sensitive to noise and thus more reliable in a real environment.

Once a threat has been detected a security robot must take some action to counter the threat. Realistically these actions will consist of raising an alarm and tracking the threat so that it is easier for human security personnel to find and neutralise it. CuWITH focuses on the latter with curiosity-driven threat tracking. CuWITH is driven to follow any curious object it has detected, how this driven movement is computed is discussed in section 4.3. Using data on the location of the object in the camera image and the area (in pixels) the image takes up on the camera image CuWITH can estimate the objects position relative to the robot and determine a driven move, either in the j-axis or l-axis, to keep the object in view.

These are essential components, however additional ancillary components are

needed for the robot to navigate and to extract objects from the camera image. These components are discussed in the next chapter

5. ANCILLARY COMPONENTS OF THE SECURITY ROBOT SYSTEM

In addition to the curiosity based threat detection and curiosity driven threat tracking components that are the focus of this research, there are several ancillary components of the CuWITH security system. These ancillary components of the CuWITH, although not the focus of this research, are necessary for the CuWITH to function effectively. Of particular interest is the programmable patrol navigation component and the object extractor. Both of these components are crucial if the CuWITH is to be effective as a research platform for curiosity based threat detection and curiosity driven threat tracking.

5.1 Programmable Patrol Navigation

One of the most important ancillary components of the CuWITH system is the programmable patrol navigation component. The major responsibility of the navigation component is twofold; to navigate the robot correctly through the patrol path and to quickly execute the movement instructions of the curiosity driven threat tracking component. A key advantage of this component is that it is programmable - the principle idea of programmable patrol navigation is that the patrol path is defined separately from the security system, in the form of a script which is supplied to the system at runtime. This script is separate from the main system and thus can be modified without having to change the navigation program. This allows for greater flexibility when testing the CuWITH under different conditions and in different situations. Because the path is programmable it is easy to modify to fit any office environment.

5.1.1 Motion Control for the CuWITH

Programmable navigation is dependant on the CuWITH being able to determine how to move from one point to another. Determination of the robots current position is the first step to doing this. The CuWITH determines its location using a simple form of odometry. This is a technique which has been successfully used in several other robot to navigate (Borenstein et al., 1997; Borenstein & Evans, 1997; Chenavier & Crowley, 1992). If the CuWITH moves left at a speed of 100mm/s for 1s then CuWITH assumes it has moved 100mm to the left. However to move from point to point CuWITH must not only move between patrol path point but must also turn to face the next point in its patrol. During a patrol, the only information CuWITH has about its position is the last patrol point it visited (point $A = A_x, A_y$), the patrol point it is currently located at (point $B = B_x, B_y$) and the next patrol point (point $C = C_x, C_y$). Using this data the CuWITH must compute both the turn magnitude θ and the direction τ necessary to turn to face point C. The magnitude of the turn needed to face point C is determined using three values derived from these points - the euclidean distance between points A and B (d_{ab}), B and C (d_{bc}) and between C and A (d_{ca}) (see Fig. 5.1). Points A,B and C form a

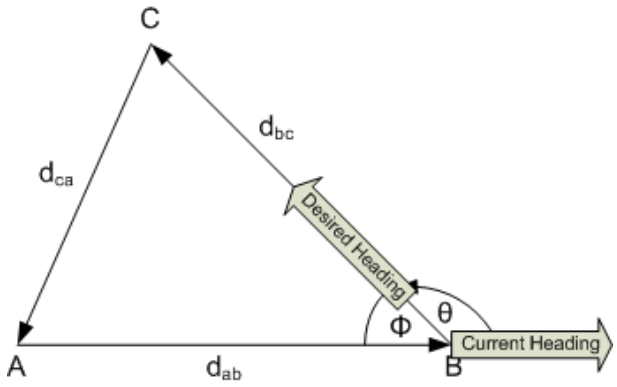


Fig. 5.1: The turn θ CuWITH is required to execute to move to point C, after moving from point A to point B

triangle. This can be taken advantage of to determine the magnitude of the turn, using simple trigonometry to determine the value of ϕ,

$$\phi = \arccos \frac{d_{ab}^2 + d_{bc}^2 - d_{ca}^2}{2.d_{ab}.d_{bc}}. \tag{5.1}$$

Since ϕ and θ are supplementary angles, determining the value of θ is a simple case of subtracting ϕ from π.

For determining the direction of this turn, there are eight different situations the robot will need to contend with, as seen in Fig. 5.2.

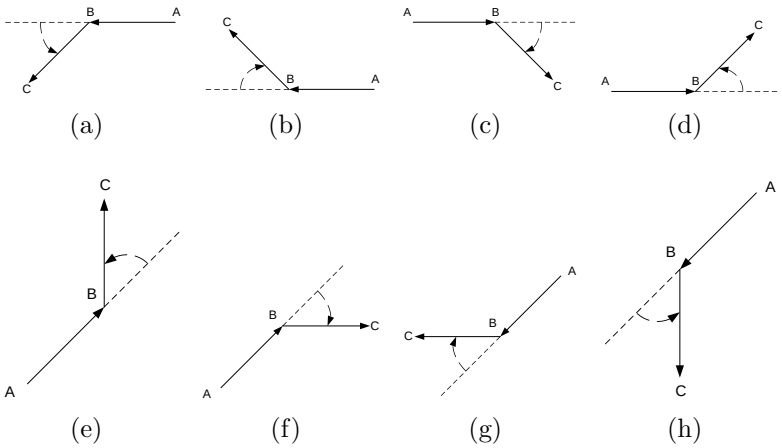


Fig. 5.2: The 8 figures represent different turning situations the robot could encounter whilst patrolling. In each diagram the robot has moved from point A to point B and needs to turn to face point C.

In the simplest case, $A_y = B_y$ but $A_x \neq B_x$. In this case there are four possible situations as seen in Fig. 5.2a – 5.2d. The direction here is determined by two factors - whether C_y is higher or lower than B_y, and whether A_x is higher or lower than B_x. The direction of the turn with these situations is determined as follows

$$\tau = \begin{cases} \text{anticlockwise} & \text{if } C_y > B_y \text{ and } B_x > A_x \\ \text{clockwise} & \text{if } C_y < B_y \text{ and } B_x > A_x \\ \text{clockwise} & \text{if } C_y > B_y \text{ and } B_x < A_x \\ \text{anticlockwise} & \text{if } C_y < B_y \text{ and } B_x < A_x \end{cases} \tag{5.2}$$

A more complex case is applicable if $A_y \neq B_y$ and $A_x \neq B_x$. In this case then the remaining four situations Fig. 5.2e – 5.2h apply. Which situation is used is dependent on two different factors. The first is the difference between A_y and B_y. The second is based on the line defined by points A and B. The gradient k and y-intercept e of this line are defined as follows:

$$k = \frac{B_x - A_x}{B_y - A_y}, \tag{5.3}$$

$$e = B_x - kB_y. \tag{5.4}$$

A point $P = P_x, P_y$ is defined on this line, with $P_y = C_y$ and P_x found as follows:

$$P_x = kC_y + e. \tag{5.5}$$

The second factor that determines which situation is the difference between P_x and C_x. This is used to determine if C_x is 'above' the AB line or below. The turn direction (assuming case Fig. 5.2e – 5.2h) can therefore be determined using the following equation.

$$\tau = \begin{cases} \text{anticlockwise} & \text{if } (C_x - P_x) > 0 \text{ and } (B_y - A_y) > 0 \\ \text{clockwise} & \text{if } (C_x - P_x) \leq 0 \text{ and } (B_y - A_y) > 0 \\ \text{clockwise} & \text{if } (C_x - P_x) > 0 \text{ and } (B_y - A_y) \leq 0 \\ \text{anticlockwise} & \text{if } (C_x - P_x) \leq 0 \text{ and } (B_y - A_y) \leq 0 \end{cases} \tag{5.6}$$

Using these situations it is possible to reliably determine the magnitude and direction of any turn regardless of the relative positions of point A,B and C.

5.1.2 An Example of Programmable Patrol Navigation

Programmable patrol navigation has been implemented in the CuWITH security system using a patrol path script file. This consists of a list of patrol points that the robot is required to visit, in order. The environment is represented as a 2-D plane, each patrol point is a pair of co-ordinates on this plane.

The patrol design system takes these points and calculates a series of instructions for the WITH robot to move it in a loop, from its current patrol point to the next patrol point, until the robot has reached the last patrol point. The robot then moves back to the first patrol point and restarts the patrol. For example, say the robot needs to move to three security points in a triangular orientation as in Fig. 5.3. To effect this a simple patrol script file as seen in Fig. 5.4 can be written. The script informs the robot that the patrol starts at security point A $(0, 0)$, the second security point B is located at $(0, 900)$ and the position of the third and final security

point is at $(450, 900)$.

This script is translated by the navigation system into robot instructions as the robot patrols. An example of the low-level instructions that may be computed over the course of the patrol can be seen in Fig. 5.4. These instructions first instruct the robot to move forward at a velocity of 100mm/s for 9000ms, bringing it to security point B (the robot is always assumed to start at the first patrol point, facing the second patrol point). Next the robot turns at a rate of 1000millirad/s for 1525ms, causing the robot to face patrol point C as instructed by the script. These instructions continue on, instructing the robot to move to security point C, then to security point A. Finally the robot turns to face security point B, ready to start the patrol again.

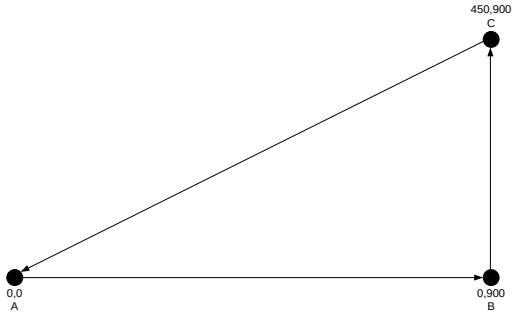


Fig. 5.3: Example patrol path for the CuWITH

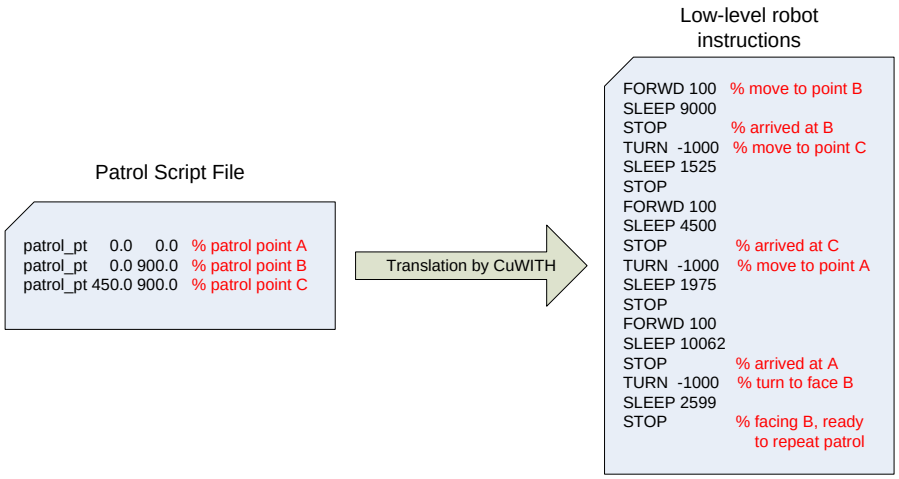


Fig. 5.4: Procedure for determining robot instructions to execute the patrol path in Fig. 5.3

5.1.3 Advantages of Programmable Patrol Navigation

Programmable patrol navigation improves the maintainability of the CuWITH and helps protect it from human error.

The patrol path of the CuWITH is defined as a simple script, defining a series of patrol path points. What improves the maintainability of the CuWITH however is that this patrol path script can easily be modified to suit the current environment of the robot. Changes to the structure of the environment, improvements in the design of the patrol path, moving the CuWITH to a completely different environment - all these situations can be easily dealt with though simple modification of the CuWITH patrol script. Furthermore these changes do not require the CuWITH system proper to be modified. Such modifications, if necessary, run the risk of introducing bugs into the system, potentially causing the robot to fail and leave the environment vulnerable. Programmable patrol navigation avoids these risks and makes maintaining the CuWITH much easier and quicker.

Programmable patrol navigation also protects the CuWITH from human error because the path is defined at a relatively high level. The patrol path designer does not have to specify how exactly to move between two points, this can be determined by the CuWITH. All that is necessary is the location of the points. If specifying the details of the movement was necessary, there is a great risk of human error particularly with more complex patrol paths. It is very easy for typos to slip in or instructions to be left out, resulting in the robot being supplied with an incorrect patrol path. This can invalidate experiments and could damage the CuWITH if it was accidentally instructed to collide into (or off) objects in the office. Programmable patrol navigation prevents this situation from happening.

As can be seen, implementing programmable patrol navigation using a script file makes patrol paths for the robot easier to implement and modify, and protects the system from the introduction of bugs through program modification. Another important ancillary component is the object extractor

5.2 Object Extraction Method

The CuWITH security system uses curiosity - particularly face curiosity - to detect threats in the environment, and is driven to follow the threat while taking some appropriate action. As part of this the robot needs to take an image from the camera, extract images of any objects (e.g. faces) from this image and submit these images for curiosity evaluation. Object extraction here is a key step - it must be reliable, fast and have low memory and computational requirements. If it is not reliable then it may submit a image which is not a face at all, resulting in a high curiosity value for a object that does not exist. If it is not fast it will not be able to keep up in a real environment. And of course if it has high memory of computational requirements it will not be possible to implement on the limited memory and computational resources of the CuWITH robot. Another consideration is that the focus of the CuWITH project is on curiosity-driven threat detection and tracking, with some work done on programmable navigation. Attempting to develop a completely new method of object detection is not within the scope of this project currently.

It was with these points in mind that the current object detector was designed. This object detector uses a object detection algorithm proposed by P. Viola et all (Viola & Jones, 2001) and improved by R. Lienhart et all (Lienhart & Maydt, 2002). This is a very fast an accurate algorithm which was implemented based on code taken from the OpenCV documentation on object detection. The original OpenCV code has then modified to extract any faces from the camera image as a face image. Fortunately trained classifiers for face detection come with the OpenCV distribution so there was no need to spend weeks training the object detector.

In practical experiments, with the programmable patrol navigation, curiosity-based threat detection and curiosity driven threat tracking all running, the object detector showed itself to be fast and accurate in a real environment. The rare misclassification can be filtered out automatically by the curiosity calculation, as the curiosity calculation excludes outliers as part of measures to improve curiosity stability.

6. EXPERIMENTS AND SYSTEM EVALUATION

To evaluate the CuWITH and our curiosity based security methods, a number of tests have been performed. There are three elements of CuWITH which have been individually tested to evaluate their effectiveness. For CuWITH to effectively patrol an office the programmable patrol navigation must be both reliable and flexible. A series of tests have been performed to determine if this is the case. However the focus of the CuWITH is curiosity-based threat detection and curiosity driven threat tracking, to that end both are thoroughly tested for their speed and accuracy. The CuWITH as a whole has also been tested to determine how well these components can work together to protect an office.

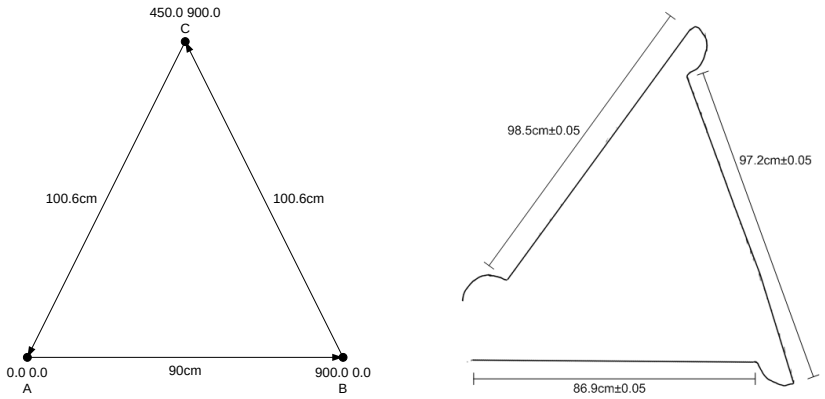
6.1 Results of the Programmable Patrol Navigation Testing

CuWITH has been designed to not only patrol a particular path but also to utilise programmable patrol navigation, allowing it to patrol any arbitrary path supplied to it without needing to change any part of the CuWITH system. There are two key aspects of this that need to be tested; first the CuWITH must correctly patrol a given path and it must be possible to change this path without any change to the CuWITH system - only the script describing the patrol path should be modified.

As an initial test a simple triangular patrol path was designed, as seen in Fig. 6.1a. This path consists of three patrol points, with the CuWITH starting at point A (0,0), moving to point B (900,0) then to point C (450,900) before returning to point A. This path was specified in a patrol path script (see Fig. 6.1a) which was supplied to the CuWITH at the beginning of the test.

During the test the CuWITH performed the patrol on a table covered in paper. A pen was attached to the rear of the CuWITH to draw the robots path during

the test. The resulting line was photographed after the test was completed. An enhanced version of this image can be seen in Fig. 6.1b.



(a) (b)

Fig. 6.1: The simple patrol path that was used to test the CuWITH's programmable patrol navigation can be seen in the left figure. The actual path CuWITH took when it performed this patrol path can be seen to the right

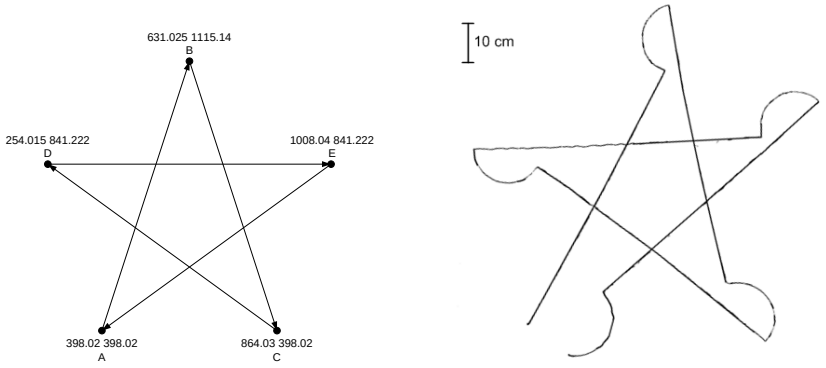
```
patrol_pt   0.0     0.0
patrol_pt   900.0   0.0
patrol_pt   450.0   900.0
```

Fig. 6.2: Patrol path script to execute the designed patrol path in Fig. 6.1a

The key question here is did the robot move correctly? This question can be answered by comparing the desired path with the actual path. The first move executed by the CuWITH is a straight move to point B. Although this move is 3.1cm shorter than the ideal distance this is not a significant deviation. A more significant deviation occurs when CuWITH turns to face point C, the turn is too small. When the CuWITH completes the following move to point C it actually ends up slightly above and to the right of the ideal position. This is a larger deviation but still reasonably close to the desired location. Finally, the CuWITH turns and moves back to point A, however it halts a little too early and ends up above and to the right of the endpoint. Comparing the designed path (Fig. 6.1a) to the actual path (Fig. 6.1b) it is clear that the CuWITH was quite close to the desired path, visiting each point in turn then returning to the origin.

This initial test however uses a simple patrol path, a triangle. In a real setting the patrol path will be more complex than this. To more thoroughly test the CuWITH's ability to handle more complex paths a star-shaped patrol path was designed (see

Fig. 6.3a). This is a star-shaped path that tests the CuWITH's ability to both move precisely and compute the correct turn direction and magnitude. The initial test was then repeated using the complex patrol path rather than the simple patrol path. The path the CuWITH took can be seen in Fig. 6.3b.



(a) (b)

Fig. 6.3: The left figure is the more complex patrol path design that was used to further test the reliability and robustness of the CuWITH's navigation. The actual path taken by CuWITH when performing this patrol path can be seen to the right.

```
patrol_pt   398.02    398.02
patrol_pt   631.025  1115.14
patrol_pt   864.03    398.02
patrol_pt   254.015   841.222
patrol_pt  1008.04    841.222
```

Fig. 6.4: Patrol path script to execute the designed patrol path in Fig. 6.3a

As can be seen the CuWITH followed the path very well, reaching every patrol point and returning to the initial position correctly. However there are still some small deviations from the ideal path. To more closely examine this error an experiment was performed to determine if this error was precisely predicable (and therefore could be easily compensated for), the experimental setup can be seen in Fig. 6.5.

In the experiment the CuWITH was faced at a pre set initial position on a table covered in paper (as with the previous experiments). The CuWITH was then instructed to move forward 90cm in a straight line. This was not performed using the navigation component of the CuWITH, to exclude the possibility of program errors a very simple control program was written which only instructed the robot to move forward exactly 90 cm. After the CuWITH had finished moving, the optimal final position of the CuWITH (assuming it had moved exactly 90cm forward in a

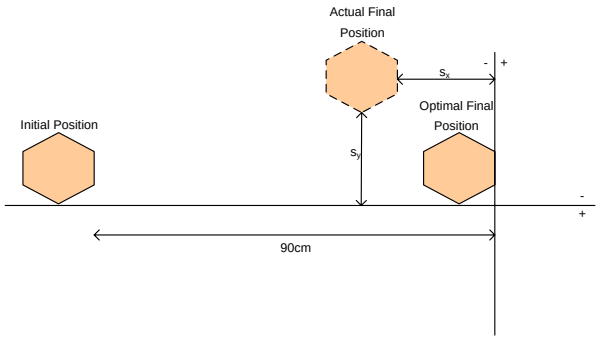


Fig. 6.5: Experimental design for the movement reliability tests. The CuWITH starts at a pre-set initial position and is instructed to move forward 90cm. Two values are measured to determine the CuWITH's deviation from the path, the x and y difference (s_x, s_y) between the robots actual final position and its optimal final position

straight line) and the actual final position of the CuWITH were compared. Two values are key, the horizontal displacement error s_x (difference between the actual distance moved forward and 90cm) and the side displacement error s_y (how far to the left or the right CuWITH deviated from the planned straight path). This experiment was repeated 50 times, the results of 48 of these experiments can be seen in Fig. 6.6 (two experiments had values for s_x that were considerably higher than the rest of the data and were excluded as outliers).

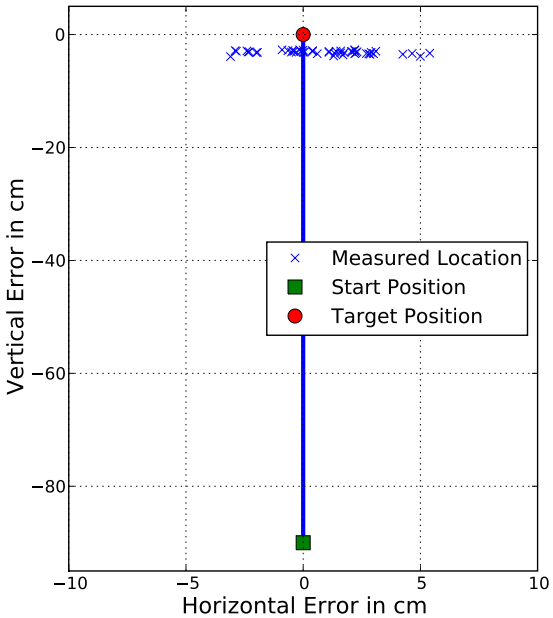


Fig. 6.6: Results of the path deviation experiment. Each point represents one experiment, 48 experiments are represented in the graph (two outliers were excluded)

The first thing that is clear from this data is that at no time did the robot reach the optimal final position, although sometimes s_y is zero CuWITH is always at least -2.7cm short of the target 90cm. There is also considerable variability in the deviations. Despite each experiment being performed the same way, CuWITH experienced a wide array of deviations, both to the left and to the right. The variation is not systematic or predictable enough to be compensated for without additional information about the CuWITH's position.

Despite these errors the fact remains that the CuWITH was able to complete both a simple and a more complex patrol path without any system modification required. For the purpose the CuWITH navigation component is intended to serve (e.g. facilitation of curiosity based threat detection and curiosity driven threat tracking experiments) the current performance is satisfactory.

6.2 Evaluation of Threat Detection

There are two key aspects of real time threat detection that curiosity based threat detection should be evaluated by. The first is the reliability of curiosity based threat detection in a real office environment. While patrolling an office environment, a security robot may have to deal with differing lighting conditions, a variety of different object poses and the distance between the robot and an object may change over time. When evaluating curiosity based threat detection these confounding factors must be considered. Threats must also be detected in real time, if this is not possible then intruders could escape detection and compromise the security of the office. Therefore when evaluating curiosity based threat detection the speed of the computation must be considered. Furthermore, an autonomous robot may not have the computational resources available to a desktop computer. To properly examine the speed of execution, curiosity based threat detection should be performed using the more limited computational resources of (in this case) the CuWITH, to more accurately represent the real world demands of a autonomous security robot. Taking these aspects into account, a number of experiments have been conducted to evaluate the effectiveness of curiosity based threat detection in real time threat detection.

6.2.1 Offline Threat Detection Experiments

Before testing curiosity based threat detection in an online experiment on the CuWITH, an offline experiment was performed using two face databases, the Lab face database and the BioID face database (*BioID Face Database*, n.d.).

The Lab face database consisted of the faces of 7 different people present in the KEDRI offices. All of these face were obtained in the office, using the CuWITH's camera, with the person seated in front of the camera. These people were lit from a fluorescent light above and to their left. The face images were then extracted from the camera image using the CuWITH's object extractor. The BioID face database (*BioID Face Database*, n.d.) is a large dataset of 1521 images, each of which shows the frontal view of the face of one of 23 subjects. The BioID database is focused on real world testing, to that end the images have been taken in a variety of illuminations, backgrounds and poses.

These databases are divided into two datasets, a training dataset and a much larger testing dataset. The training dataset contains data on 5 people represented in the Lab face dataset, for each person a small subset of their faces in the Lab face database were included in the trained dataset. The remaining face images from both the Lab face database and the BioID face database was then included in the test database. Furthermore, to test how robust curiosity based threat detection was to differing numbers of known people, all tests were repeated using 4,3,2 and 1 people in the training database.

The experiment was performed by first training the curiosity based threat detection component using the training dataset, then determining the curiosity value of each of the faces in the testing dataset (the number of eigenvectors $M' = 20$). If the square root of the curiosity value of a particular face was higher than a threshold (70) then that face was designated as curious and a threat, otherwise the face was known and not threatening. This result was then compared with the true threat status of the face to see if this classification was accurate. The results of these tests can be seen in Table. 6.1.

As can be seen, the percentage of threatening (unknown) faces correctly regarded

Tab. 6.1: Results of the Offline Experiments

known people	N_{train}	test faces (known/unknown)	T_p Rate	T_n Rate
5	1250	5479 (3617/1862)	83.73 %	85.84 %
4	1000	5230 (3368/1862)	84.75 %	84.76 %
3	750	4724 (2862/1862)	85.45 %	99.65 %
2	499	3032 (1170/1862)	85.61 %	99.915 %
1	248	2356 (494/1862)	89.31 %	100.0 %

as threats ranges between 83% and 89%, while the percentage of known faces which were not regarded as threats is above 99% for three fifths of the tests. Although both these numbers are lower than our best expectations, we believe this is due to PCAs sensitivity to changes in light, viewing angle and face size. This is a well known limitation of PCA (Turk & Pentland, 1991; Zhang et al., 1997). Despite this PCA has a distinct advantage in its low computational requirements. The robots VGN-UX70 minicomputer has relatively low computing power, a very accurate algorithm would exceed its capabilities and not run in real time. As will be demonstrated in the next section, PCA not only can run in real time on the minicomputer but can also attain good accuracy.

6.2.2 Online testing in an Office Environment

To test the real-time performance of curiosity based threat detection the accuracy and frame rate of CBTD was tested in a real environment. This environment was a room in the KEDRI offices and consisted of the CuWITH on a table facing a subject seated in front of a bookcase (see Fig. 6.7). The subject was lit by a light above and to the left of the subject, as well as by several more distant lights.

Training data was collected in this environment, using the CuWITH to obtain face images of two people in this office environment, located in the same position as the unknown person in 6.7. The CBTD was then trained on this data and tested using two different test subjects. One of these was a known person (i.e. one of the people the CBTD had been trained on), the other was a person who was unknown to the CuWITH. Each test subject was placed in front of the CuWITH (as seen in Fig. 6.7) and the computed curiosity values for that person were recorded over 60

Fig. 6.7: The environment of the subjects when obtaining the test data. In this example an unknown person is present in the environment.

seconds.

The first experiment concerned the easiest case, with the subject looking directly at the camera. In this experiment 753 and 756 frames were processed for the known and unknown person respectively. The curiosity values for each frame processed during this experiment can be seen in Fig. 6.8.

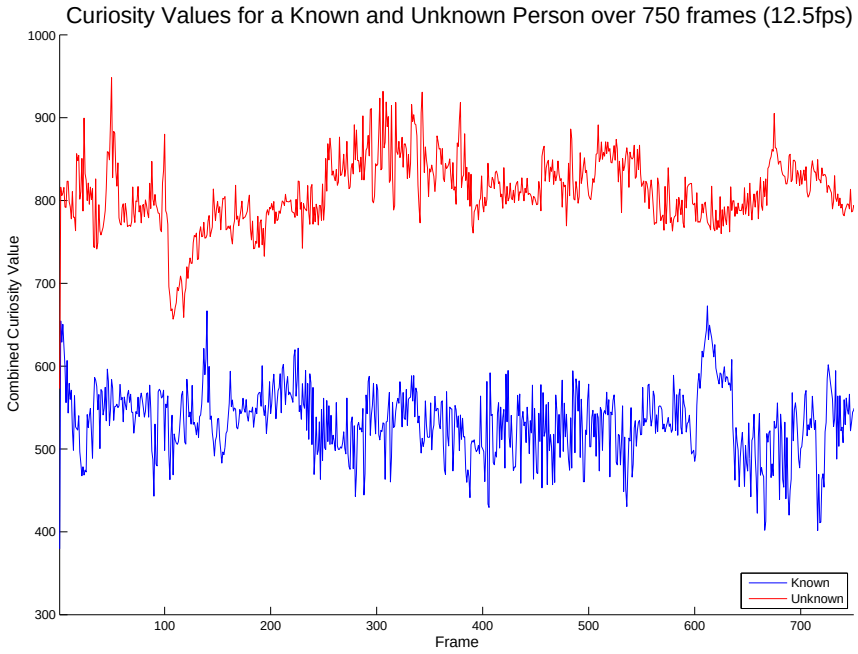


Fig. 6.8: The results of the initial curiosity experiment. The x-axis of this graph shows the frame count (each frame is captured and processed by CuWITH in real time). The y-axis displays the raw curiosity value computed for each frame for both the known and unknown subject

When judging how effective the curiosity computation is, the major consideration is separability - can the unknown and known person be differentiated, and if so how significant is the difference between the two. It is also expected that the curiosity values of the unknown subject are higher than the curiosity values for the known subject - the unknown subject should be much more curious than the known. Curiosity should also be calculated in real time, otherwise the CuWITH may miss important security events occurring in the environment.

The test results for both the known and unknown person were collected over the course of 60 seconds. 753-756 frames over 60 seconds means that the CuWITH's CBTD achieved a frame rate of 12.575 fps ± 0.025. This is sufficient for real-time processing, demonstrating not only that the CuWITH is capable of reliably detecting threats in a real environment but also that it can do this in real time.

As can be seen, the curiosity values for the unknown person (shown in red) are almost all ≈ 100 above the curiosity values for the unknown person (shown in blue). However there are some fluctuations in the values, and some of these fluctuations are severe enough that the curiosity value gap between the known and unknown people disappears. For example, frame 108 of the unknown person has a curiosity value of only 656.7, while the curiosity value of the known person is as high as 668.8 at frame 140 and 672.8 at frame 612. The severity of these fluctuations can be reduced by evaluating the curiosity values over some time period κ rather than individually.

In CuWITH this has been implemented by examining the values over a time period $(t - \kappa)...t$ to determine at time t if a threat is present, as seen in Eq. 4.8. Each curiosity value in the group is evaluated as either unknown or known, if the percentage of unknown evaluations is higher than a threshold a threat is detected. Noise in the data being acquired can therefore be marginalised - if the majority of curiosity values are relatively low (for example if only a known person is present) a single abnormally high value cannot overshadow this and cause a false positive detection.

To demonstrate this let us take the data presented in Fig. 6.8 and divide it into sequential groups of ten. Each group of ten is displayed as a box plot in Fig. 6.9,

numbered in time order.

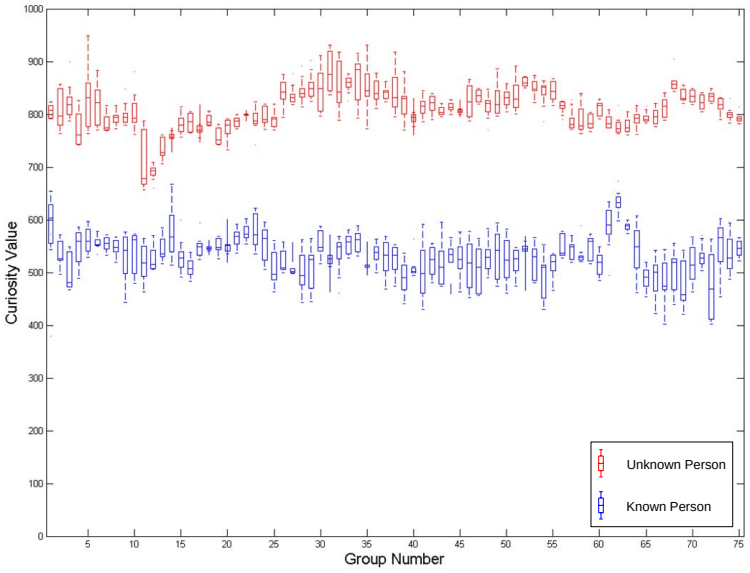


Fig. 6.9: The results of Fig. 6.8 when the spread over 10 sequential curiosity values is considered. The y axis displays the curiosity value while the x-axis displays the frame count. Each box plot displays the spread of the previous 10 curiosity values, with the box being bounded by the 75th and 25th percentiles.

The box of each box plot in Fig. 6.9 stretches between the 75th and 25th percentiles. Note that for the unknown person the 25th percentile is almost always higher than 700, and even the two exceptions to this are higher than 650. This means that for the majority of the unknown groups at least 75% of the curiosity values are higher than 700. Conversely with the known person the 75th percentile of the majority of the groups are below 625. What is key to note is that the difference between the 25th percentiles of the unknown person and the 75th percentiles of the unknown person is realtively large - in most cases it is larger than 75. This clearly shows that there is a reliable difference between the known and unknown person. With such a gap, it is possible to reliably detect threats - a threshold can be set which will reliably classify most known face images as known, and most unknown face images as unknown.

The results for this base experiment are encouraging, however they assume a best case scenario where the person will be looking directly at the camera. In reality this cannot be relied on so several other test cases were used to evaluate the reliability of the CuWITH's CBTD under different subject poses. These test cases involved the subject facing to the left and right (from the CuWITH's perspective) and the

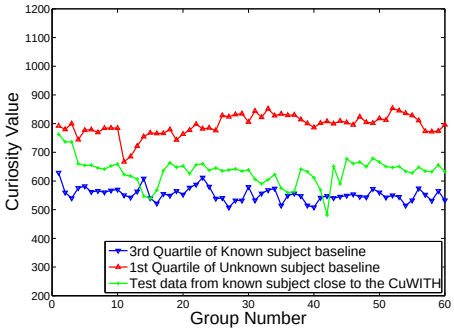
subject being significantly closer and further away from the CuWITH.

The experimental setup was the same as the initial experiment, with the subject sitting before the CuWITH for 60 seconds, however in this instance the subject assumed one of the aforementioned poses. Some data was lost here as the different poses (particularly the poses that had the subject facing left or right) occasionally caused the face detection to fail, but still a significant amount of data was obtained.

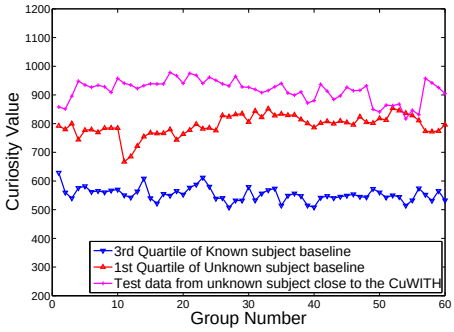
The data collected through these experiments was compared to the baseline data to see how the different poses affected the performance of CBTD. More specifically, the most important values of the baseline were used for comparison - the 25th percentile of the unknown baseline and the 75th percentile of the known baseline. This was compared to the 75th percentile of the known person test data and the 25th percentile of the unknown person test data. This can be seen in Fig. 6.10

The first series of experiments are the 'close' experiments (Fig. 6.10a and 6.10b), where the subject was significantly closer to the CuWITH than with the baseline data. The combined curiosity values for the unknown subject when they are close to the CuWITH (Fig. 6.10b) are clearly slightly higher than the unknown subject baseline. This is in line with what is required of the CuWITH's threat detection - despite the change in pose the curiosity value for the unknown subject is within acceptable boundaries. The CuWITH's curiosity computation does not perform as well for the known subject when they are close to the CuWITH (Fig. 6.10a). The computed curiosity values are slightly higher than the known subject baseline, however the values are still below the unknown subject baseline by at least ≈ 20, with the majority of the values having a much higher difference. The first 33 frames are an exception to this, with known test values as high as the unknown subject baseline. Despite this the majority of test values are still below the unknown subject baseline, allowing for a threshold to be set with reasonable room for error.

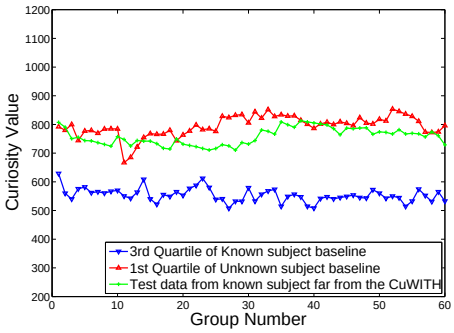
The second series of tests examine CuWITH's performance against subjects which are relatively distant from the robot (Fig. 6.10c and 6.10d). The unknown subject is the case that is handled best, with combined curiosity values well above the unknown subject baseline. The case of the known subject however performs very



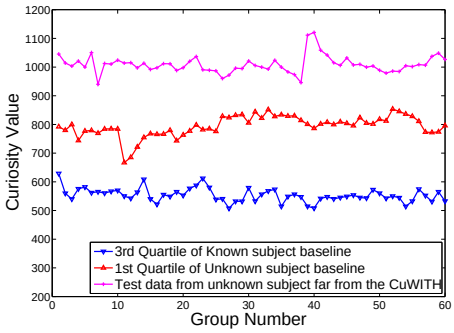
(a) Known person close to the camera



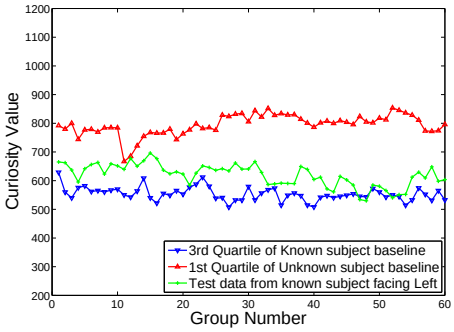
(b) Unknown person close to the camera



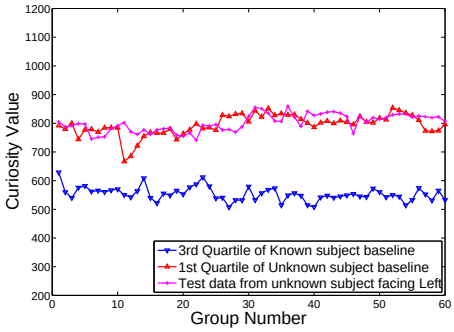
(c) Known person far from the camera



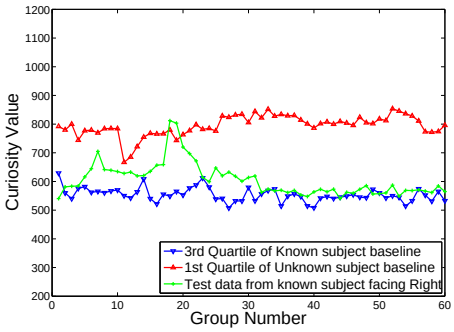
(d) Unknown person far from the camera



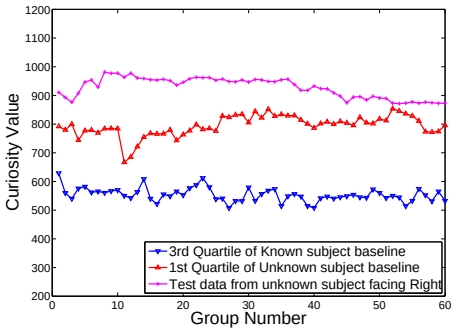
(e) Known person facing left



(f) Unknown person facing left



(g) Known person facing right



(h) Unknown person facing right

Fig. 6.10: Comparison of the CuWITH's threat detection performance under different subject poses and locations.

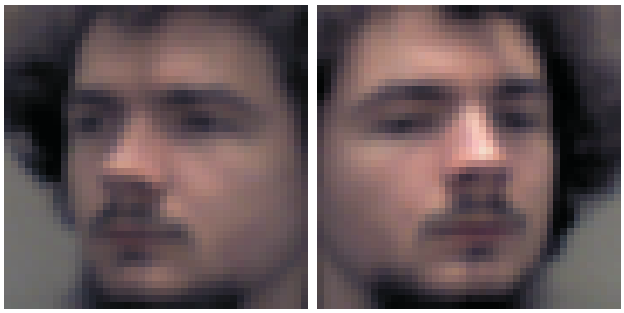
poorly, with the curiosity values consistently within the same range as the unknown subject baseline. It is not possible to determine a threshold which could differentiate between the known subject in this case and the unknown subject baseline.

The third series of tests observes the reliability of the CuWITH's combined curiosity calculation when the subject is facing to the left of the CuWITH (Fig. 6.10e and 6.10f). In this case the performance of both the known and unknown subject is very good. The known subject test values correlate very well with the known subject baseline, and similarly the unknown subject test values are all within the range of the unknown subject baseline.

The final series of experiments examines how well the CuWITH performs when the subject is facing to the right of the CuWITH. The unknown subject is handled well, with the curiosity values being equal to or higher than the unknown subject baseline. The known subject is not handled as well, the values are very close to the unknown baseline. A threshold can still be applied as there is still a gap between the two sets of combined curiosity values, however it will not be as reliable.

When considering these results there are two unusual trends which can be observed. The first is that the curiosity values of the subject when they are facing the CuWITH's right side are higher than when they are facing the CuWITH's left side. This is a significant difference which is too large to be caused by the left side of the subjects face being different from the right side. The answer lies in the environmental conditions of the experiment; there is a light above *and to the left* of the subject. This difference of lighting causes the right side of the face to be less brightly lit than the left. When the subject faces to the robots left this is not as much of a problem, the face is lit much the same as when the subject is facing the CuWITH. However when the subject is facing in the other direction the face is more brightly lit. For example, the 201st frame of the known subject facing right results in the highest curiosity value of all the frames in that experiment (see Fig. 6.10g). If one compares this frame with the 201st frame from the left side experiment a clear difference in lighting can be seen (Fig. 6.11).

As can be seen the right side face is more brightly lit than the leftside face. there



(a) (b)

Fig. 6.11: Comparison of a known person facing to the left and to the right of the robot. As can be seen, the image on the right is more brightly lit than the image to the left, despite the fact that the subject and the CuWITH were in the same location for both images

is also a greater variance in the lighting on the right side face - while the left side face is lit more uniformly, the right side is more varied, with some areas of the face being more dark and others being more light. I believe it is this which is causing the right side experiments to report higher curiosity values than the left side experiments.

The second trend that can be seen from these results is the poor results of the far experiment with the known subject. I believe the most likely explanation is that as the subject recedes from the CuWITH, the area of the camera image that the face takes up decreases. This means that the CuWITH has less information about the face compared to what the baseline experiment (where the subjects face was closer to the CuWITH) had. This is a well documented limitation of PCA (Turk & Pentland, 1991; Zhang et al., 1997).

It should be noted that for these tests only a single eigenvector was used, while this would have made the computation faster it also had severe drawbacks. Although two people were used to train the curiosity model, only one could ever be correctly classified as known. We believe this is because only one eigenvector was not enough to discriminate between these two people, only one or the other could be learned.

Alongside the CBTD of the CuWITH the curiosity driven threat tracking is another important component of the CuWITH which needs to be tested.

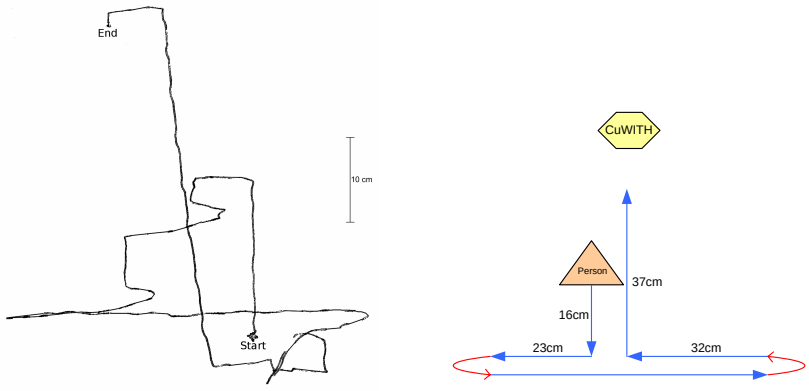
6.3 Evaluation of Curiosity Driven Threat Tracking

Detection of threats is crucial for effective robot security, however the actions taken when a threat is detected are just as important. In the case of CuWITH, it is driven by its curiosity of unknown objects to follow threats as discussed in section 4.3. This is necessary for the CuWITH to protect its environment effectively; it must keep track of where the threat is to facilitate threat neutralisation.

To test the accuracy and speed of the curiosity driven threat tracking of the CuWITH a experiment was performed. This test consisted of CuWITH being shown a person, who then moved in a particular pattern. The CuWITH's path was then recorded to observe if it was capable of following the person correctly. This experiment was performed in a real environment, with the subject initially sitting in a chair in front of the CuWITH, with a bookcase behind them.

The movement of the person can be seen in Fig. 6.12b. In this diagram the blue lines represent the movement of the person, the red lines simply clarify the order in which the movements are executed. As can be seen, the person first moves backward, then to the right, to the left, back right to the centre then moves forward past the starting location.

The real path of CuWITH in response to this movement can be seen in Fig. 6.12a.



(a) The path of CuWITH when following the unknown person

(b) The path of the unknown person during the threat tracking experiment

Fig. 6.12: Results of the threat tracking experiment. The right figure shows the actual movement of the unknown person, the left figure illustrates how the CuWITH moved to follow the persons movements

Initially the robot moved forward rather than backward, this is because the subject was too far away from the robot and thus caused the CuWITH to be driven forward. After an appropriate distance between the subject and the CuWITH was attained the test proper began. The subject first moved backwards. As can be seen in Fig. 6.12a this triggered a response by the CuWITH, with the robot moving forward following the person. CuWITH also moves side to side, this is because the subject moved to the left or right slightly while moving backward. CuWITH then continues to follow the face, moving to the left, then the right, back to the left then following the person as they move forward. Thus the CuWITH is able to follow the person accurately as they move in the environment.

This test also provided an opportunity to measure the reaction time of the CuWITH. By recording the above test using a separate fixed camera it is possible to estimate how quickly the CuWITH can react to a changing environment. This reaction time can be seen in the sequence of frames shown in Fig. 6.13:

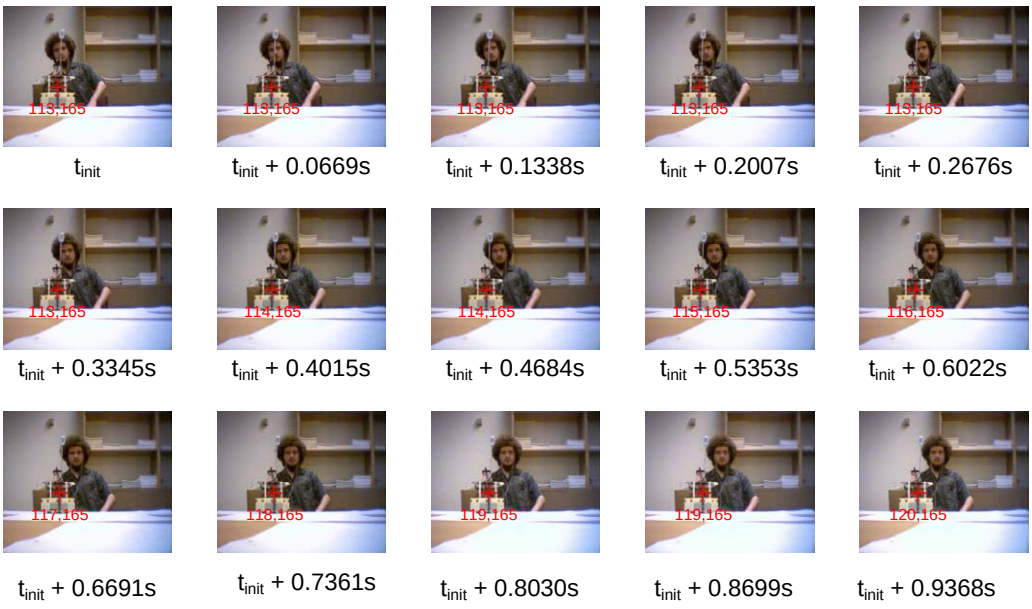


Fig. 6.13: This series of frames shows the reaction time of the CuWITH. Each frame has a cross (and a set of x,y coordinates) marking the location of the CuWITH. At t_{init} the person begins to move to the CuWITH's right, 401.5ms after the person starts to move CuWITH detects the movement and begins to follow the person to the right

Initially (time t_{init}) the person is within the centre of the CuWITH's view. As the person moves to the CuWITH's right the CuWITH does not react at first,

primarily because the face is still within the margin of error. At time $t_{init} + 0.4015s$ the CuWITH starts to follow the face right. This can be seen as the image of the CuWITH moves one pixel to the right to (114,165). This movement continues on though the next 7 frames, clearly showing that this is actual movement of the CuWITH rather than noise. This indicates a reaction time of 0.4015s, however the true reaction time is slightly shorter than this because some of the delay is caused by the persons face being within the margin of error.

Based on the results of these experiments it is clear that the CuWITH is capable of both detecting and tracking threats. However, one of the key advantages a mobile security robot has over a stationary camera is that it can move. A mobile security robot should be able to patrol its environment and search for threats. This capacity has been implemented in the CuWITH (discussed in Chapter 5.1) along with programmable navigation, allowing even complex patrol paths to be specified.

6.4 Unified System Testing

The most important test of the CuWITH is a unified system test that tests all components of the CuWITH working together to secure an environment. Although the individual components can perform their duties correctly in real time there are numerous issues that can come up when these components are combined. These issues include (but are not limited to) ensuring the navigation and threat detection components of the CuWITH communicate properly and whether the CuWITH has the computational power for both components to run in real time simultaneously.

To this end an environment was designed, consisting of two tables side by side covered by a tablecloth, several chairs and a bookcase as seen in Fig. 6.14.

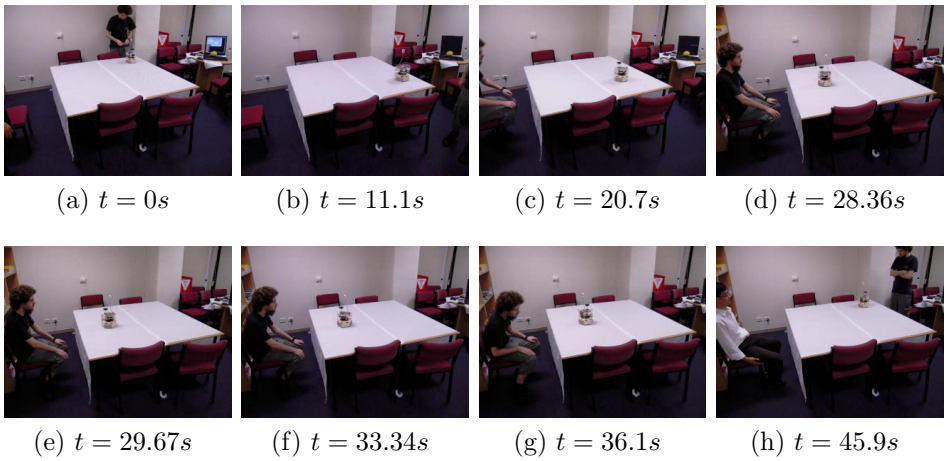
A patrol path was then designed for the CuWITH. This patrol path was a triangular patrol path (similar to the triangular path in Fig. 6.1a but larger). An additional step was added to instruct the robot to turn to face the bookshelf at point C. At this point CuWITH would search for threats and if a threat was detected CuWITH would be driven by its curiosity of the threat to follow it. For testing purposes an artificial timeout was added, causing the CuWITH to stop fol-

Fig. 6.14: The environment used for unified system testing

lowing the threat after a set amount of time had passed. Once this timeout had been reach or if no threats where detected the CuWITH would return to its patrol. The entire experiment was recorded for later analysis.

The first time the CuWITH executed its patrol a known person was seated in front of point C. In this case the CuWITH was expected to ignore the subject and continue with its patrol. Of course the patrol itself should be executed correctly, each patrol point should be visited and the CuWITH should execute the turn at point C.

Sample frames from the first patrol can be seen in Fig. 6.15.



(a) $t = 0s$ (b) $t = 11.1s$ (c) $t = 20.7s$ (d) $t = 28.36s$

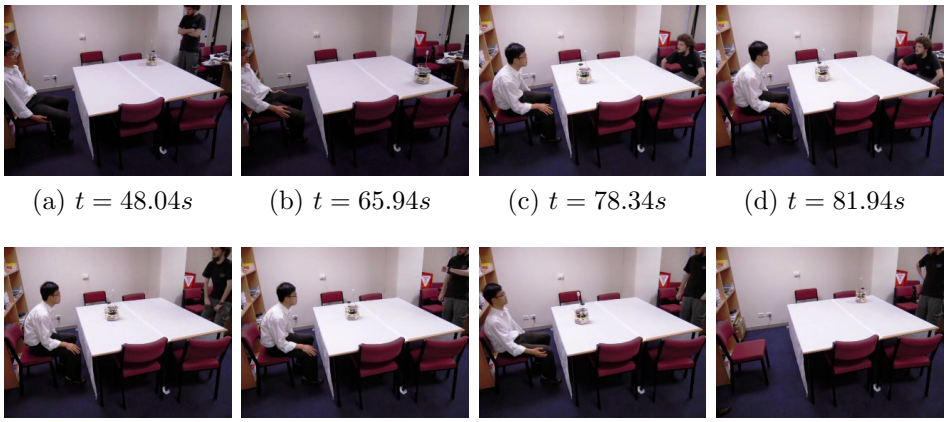
(e) $t = 29.67s$ (f) $t = 33.34s$ (g) $t = 36.1s$ (h) $t = 45.9s$

Fig. 6.15: The first patrol of the robot during the full system test. This patrol was recorded on video and screen captures at various points of the patrol are displayed here. A known person was presented to the CuWITH during this patrol (see Fig. c), as expected the CuWITH ignored this person and continued its patrol.

As can be seen CuWITH does indeed complete the designed patrol path correctly, visiting each patrol point in turn before returning to the start (Fig. 6.15g) and

turning to face point B (Fig. 6.15h). Furthermore if one examines the actions the CuWITH takes at the security checkpoint (Fig. 6.15e - 6.15g) it is clear that the CuWITH is acting correctly. It turns top face the bookshelf at point C, scans for threats with a known person before it, then ignores the known person and returns to the patrol.

The second time the CuWITH performed the patrol the known subject was replaced with an unknown subject. Sample frames from this patrol can be seen in Fig. 6.16.



(a) $t = 48.04s$ (b) $t = 65.94s$ (c) $t = 78.34s$ (d) $t = 81.94s$

(e) $t = 83.54s$ (f) $t = 84.86s$ (g) $t = 93.84s$ (h) $t = 107.34s$

Fig. 6.16: The second patrol of the robot during the full system test. As with the first patrol this was recorded on video and screen captures at various points of the patrol are displayed here. This time an unknown person was presented to the CuWITH (see Fig. c). The CuWITH proceeds to follow the person for several seconds Fig. (c - f) before returning to the patrol due to the artificial timeout

The CuWITH performs the patrol as before, however when it reaches point C and searches for threats there is an unknown person present. This indicates that a threat has been detected (i.e. the unknown person seated before the CuWITH).

At this point the CuWITH halts its patrol and starts to follow the subject, moving backward and to the right (Fig. 6.16c - 6.16f). The artificial timeout then triggers and the CuWITH returns to the patrol.

This test was repeated at the ICONIP'08 conference, with several unknown people being presented to the robot. Each person was correctly classified as a threat, even though the CuWITH was in a different environment with people that it had never encountered.

6.5 Summary

In summary, not only have the three major components of the CuWITH shown their effectiveness in performing their task under realistic experimental conditions but have shown that all three components can be combined together and run correctly on a robot in real time.

The programmable patrol navigation component has shown itself to be extremely flexible. It was able to correctly execute a relatively simple triangular patrol path (Fig. 6.1b) as well as a significantly more complex star-shaped patrol path (Fig. 6.3b). Switching from the simple patrol path to the complex patrol path requires only a simple modification of the patrol path script. In both the simple and complex paths each patrol path point was visited correctly by the CuWITH, demonstrating the flexibility and accuracy of the programmable patrol navigation component.

The curiosity based threat detection component of the CuWITH has shown itself capable of detecting threats in real time whilst limited to the modest computational power of the Sony VAIO VGN-UX70 minicomputer. Initial offline tests demonstrated the potential of CBTD to reliably detect threats in a real environment. CBTD was then evaluated using online testing in a real environment. Despite displaying some sensitivity to the lighting conditions and the pose of the subject, CBTD showed an acceptable ability to reliably detect threats. Furthermore it was able to compute individual curiosity values in real time (12.5 frames processed per second).

The third and final major component, the curiosity driven threat tracking component, also demonstrated its ability to follow a moving threat in real time. Not only did it react quickly to changes in the threat's position, it was also able to move accurately and match the threats motion in real time. All of this was achieved under true-to-life experimental conditions.

It is insufficient to show that these components are effective individually; they must be able to work together effectively, and in real time. This was first demonstrated in a full system test in a real office environment. These tests showed that each of the components were indeed working together correctly - the patrol navigation was completed successfully, the CBTD correctly ignored the known person

while detecting the unknown person and the threat tracking component was able to follow the threat as it moved in the environment. This was demonstrated again at the ICONIP'08 conference. Despite the environment being completely different the CuWITH was still able to patrol the environment and detect and handle threats correctly.

7. CONCLUSION AND FUTURE WORK

This thesis describes the development of the CuWITH, an autonomous mobile robot for office security. CuWITH implements solutions to a number of typical challenges of navigation, threat detection and threat tracking that face security robots protecting an office environment.

For such a security robot to be effective it must be able to navigate in a complex office environment. Furthermore, there are many different offices with different arrangements of corridors, rooms and doors. The developed navigation system must therefore be able to adapt to any given office. We propose the use of a programmable patrol navigation system to solve this challenge. In the programmable patrol navigation system we have developed, the patrol path is not hard coded into the system, but rather is defined in a patrol path script file that is read and executed by the system at runtime. This script is not only easy to modify to suit any given office, it also defines the path at a high level, protecting the user from making errors in defining the path. This system has been tested using the CuWITH, and was shown to work as expected by executing both a simple and a complex patrol path.

Accurate and fast threat detection is the core of the CuWITH. We propose a new concept of threat detection based on the notion of curiosity, the intrinsic motivation to investigate unexpected and unusual objects in the environment. Consider a security robot patrolling an office. The robot can expect to encounter certain objects during its patrol (e.g. particular people allowed in the office); threats to the office will be different from what is normally encountered by the robot and therefore will be curious objects. This curiosity method has been applied to face curiosity, to detect any unknown person as a threat. In offline tests, the proposed curiosity based threat detection demonstrated consistently high classification accuracy for both threats and non-threats, even when the number of different people used to

train the system varied. Further in real time experiments, the proposed method has also demonstrated its accuracy, real time execution time and robustness to different poses of the object.

Once a threat has been detected, the security robot should take some corrective action. Except for raising an alarm, threat tracking is researched as a corrective action, allowing the CuWITH to help security forces to locate and respond to intrusions. This is done using a curiosity driven approach, the CuWITH is driven by its curiosity to follow unusual objects that it detects. Experimental results show that CuWITH can react quickly and accurately to the movements of threat objects.

The effectiveness of these techniques integration for a security patrol was demonstrated in the CuWITH. In tests, the CuWITH was given three tasks: to navigate a patrol path as specified in a script file, to correctly detect a threat and to ignore a non-threat (i.e. a known person), and finally to follow the detected threat as it moved in the environment. This test was performed in two different environments, in the research offices and at the ICONIP'08 conference in Sky City Convention Centre. In both of these different environments, the CuWITH correctly executed the patrol path, detected threats while ignoring known people, and successfully tracked detected threats.

In general, CuWITH has demonstrated its ability to solve some of the challenges security robots face, and it can be further enhanced in the future. In particular there is enormous scope for future work with the CuWITH's curiosity modelling. The current model could be expanded to consider a greater variety of information, such as the time of day and the location of the threat. There is also the possibility of the CuWITH not only detecting definite threats (such as an unknown person in the office) but also possible threats. For example, a loud noise in the office could be nothing, but also could be an intruder breaking in. In this situation the CuWITH should not only detect the possible threat, but should also be driven to investigate the noise and confirm if it is a threat before raising an alarm. Such a curiosity model could also allow the CuWITH to prioritise, if the robot hears an unusual noise and detects an open door at the same time, the CuWITH will have to choose

what to investigate first. With a curiosity model that considers possible threats, it is possible to rank these possible threats (noise and open door) and decide which is more threatening. This can be combined with other information, during office hours an open door is not as much of a threat, however at night an open door may be more of a concern. The navigation system of the CuWITH could also be developed further through the use of more advanced localisation techniques to determine its location and correct its path. The camera used by the CuWITH currently can suffer from blurring if the camera or objects in view move too fast. The severity of this can be reduced by upgrading the current camera to a camera with a shorter exposure time. Nevertheless, the developed CuWITH has demonstrated its ability to perform as a mobile security agent in a office environment.

REFERENCES

Andreasson, H., Magnusson, M., & Lilienthal, A. (2007). Has something changed here? autonomous difference detection for security patrol robots. In *Proceedings of the 2007 IEEE international conference on intelligent robots and systems.*

Barreto, J., Menezes, P., & Dias, J. (2004). Human-robot interation based on harr-like features and eigenfaces. In *Proceedings of the 2004 IEEE international conference on robotics and automation.*

Barto, A., Singh, S., & Chentanez, N. (2004). Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the 2nd international conference on development and learning.*

Bioid face database. (n.d.). Website. Available from `http://www.bioid.com/downloads/facedb/index.php`

Borenstein, J., & Evans, J. (1997). The omnimate mobile robot-design, implementation and experimental results. In *Proceedings of the 1997 IEEE international conference on robotics and automation* (Vol. 4, pp. 3505–3510).

Borenstein, J., Everett, H. R., Feng, L., & Wehe, D. (1997). Mobile robot positioning: Sensors and techniques. *Journal of Robotic Systems*, *14*, 231–249.

Chenavier, F., & Crowley, J. L. (1992). Position estimation for a mobile robot using vision and odometry. In *Proceedings of the 1992 IEEE international conference on robotics and automation* (Vol. 3, pp. 2588–2593).

Chien, T. L., Su, K. L., & Guo, J. H. (2005). The multiple interface security robot - WFSR-11. In *Proceedings of the 2005 IEEE international workshop on safety, security and rescue robotics.*

Filliat, D., & Meyer, J.-A. (2003). Map-based navigation in mobile robots:: I. a review of localization strategies. *Cognitive Systems Research*, *4*, 243–282.

Huang, X., & Weng, J. (2002). Novelty and reinforcement learning in the value system of developmental robots. In *Proceedings of the 2nd international workshop on epigenetic robotics: Modelling cognitive development in robotic systems (epirob'02).*

Kim, Y.-G., Kim, H.-K., Lee, S.-G., & Lee, K.-D. (2006). Ubiquitous home security robot based on sensor network. In *Proceedings of the IEEE/WIC/ACM international conference on inteligent agent technology.*

Lee, D., Chung, W., & Kim, M. (2003). A reliable position estimation method of the service robot by map matching. In *Proceedings of the 2003 IEEE international conference on robotics and automation.*

Lienhart, R., & Maydt, J. (2002). An extended set of haar-like features for rapid object detection. In *Proceedings of the 2002 international conference on image processing* (Vol. 1).

Liu, J. N. K., Wang, M., & Feng, B. (2005). iBotGuard: An internet-based inteligent robot security system using invariant face recognition against intruder. *IEEE Transactions on Systems, Man and Cybernetics - part B: Cybernetics*, *35*.

Luo, R. C., Lin, T. Y., Chen, H. C., & Su, K. L. (2006). Multisensor based security robot system for intelligent building. In *IEEE international conference on multisensor fusion and integration for intelligent systems.*

Marshall, J., Blank, D., & Meeden, L. (2004). An emergent framework for self-motivation in developmental robotics. In *Proceedings of the 2nd international conference on development and learning (ICDL 2004).*

Mori, K., Sato, M., Sonoda, T., & Ishii, K. (2007, August). Toward realization of swarm intelligence. In *Proceedings of the 7th POSTECH-KYUTECH joint workshop on neuroinformatics.*

Oudeyer, P., Kaplan, F., & Hafner, V. V. (2007). Intrinsic motivational systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, *11*.

Pang, S., Ozawa, S., & Kasabov, N. (2005). Incremental linear discriminant analysis for classification of data streams. *IEEE Transactions on Systems, Man and*

Cybernetics - part B: Cybernetics, *35*.

Rencken, W. D. (1993). Concurrent localisation and map building for mobile robots usingultrasonic sensors. In *Proceedings of the 1993 IEEE/RSJ international conference on intelligent robots and systems* (Vol. 3, pp. 2192–2197).

Ryu, J.-G., Kil, S.-K., Shim, H.-M., Lee, S.-M., Lee, E.-H., & Hong, S.-H. (2006). Intelligence and security informatics. In (Vol. 3975, pp. 633–638). Springer Berlin/Heidelberg.

Scott, P. D., & Markovitch, S. (1989). Learning novel domains through curiosity and conjecture. In *Proceedings of international joint conference for artificial intelligence* (pp. 669–674).

Shimo, N., Pang, S., Kasabov, N., & Yamakawa, T. (2008). Curiosity-driven multi-agent competitive and cooperative LDA learning. *International Journal of Innovative Computing, Information and Control*.

Shimosasa, Y., Kanemoto, J., Hakamada, K., Horii, H., Ariki, T., Sugawara, Y., et al. (2000). Some results of the test operation of security service system with autonomous guard robot. In *26th annual conference of the IEEE on industrial electronics society* (Vol. 1, pp. 405–409).

Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, *99*, 21–71.

Treptow, A., Cielniak, G., & Duckett, T. (2005). Active people recognition using thermal and grey images on a mobile robot. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 2103–2108).

Turk, M. A., & Pentland, A. P. (1991). Face recognition using eigenfaces. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition.*

Uğur, E., Doğar, M. R., Çakmak, M., & Şahin, E. (2007). Curiosity-driven learning of traversability affordance on a mobile robot. In *Ieee 6th international conference on development and learning.*

Viola, P., & Jones, M. J. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference*

on computer vision and pattern recognition (Vol. 1).

Zhang, J., Yan, Y., & Lades, M. (1997, sep). Face recognition: Eigenface, elastic matching, and neural nets.

APPENDIX

A. GLOSSARY

PCA – Principal Component Analysis

LDA – Linear Discriminant Analysis

ILDA – Incremental Linear Discriminant Analysis

cILDA – Curiosity-driven Incremental Linear Discriminant Analysis

WITH – A small mobile robot developed at Kitakyushu Institute of Technology for use in swarm robotics (Mori et al., 2007)

CuWITH – Curious WITH, the security robot developed in this research.

Curiosity – An emotion that represents a drive to investigate and learn new things.

B. THE SPECIFICATION OF THE WITH PLATFORM

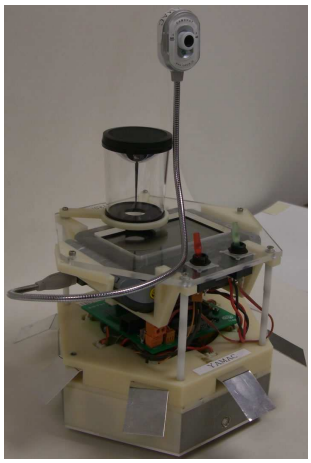


Fig. B.1: The WITH robot that the CuWITH was developed on

The CuWITH is built on top of the WITH robot developed at Kitakyushu Institute of Technology, Japan (Mori et al., 2007). Its specifications are listed in Table B.1. The WITH robot was designed as a small, mobile robot for research into swarm intelligence. It is equipped with 8 infra-red distance sensors with a range of 10-30cm, and is capable omnidirectional movement using its 3 wheels, which are arranged every 120 degrees. The WITH has been further enhanced by equipping it with a Sony VAIO VGN-UX70 minicomputer. The VGN-UX70 has built-in wireless access, considerably more computing power and memory than the WITHs control chip, and a USB port. The VGN-UX70 also has a built in camera, in combination with a mirror place directly above the camera this allows the WITH to utilise omnidirectional vision. The WITH robot used for the CuWITH is also equipped with a USBCAM30 usb camera, which was used for obtaining vision data for threat detection.

Size	Height 120mm, Width 200mm
Weight	1.5kg
Top Speed	440mm/s
Maximum Turn Speed	7.42 rads/s
Actuators	Maxon 1.2W DC-Motor x3
CPU	dsPIC30F6014 (Main CPU) dsPIC30F2010 x3 (Motor Controllers)
Communication	I2C (400kbps max) UART (1.2Mbps max) Wireless LAN (115.2kbps max)
Sensor	Infrared Distance Sensor x8
Installed Minicomputer	Sony VAIO VGN-UX70
camera	USBCAM30

Tab. B.1: WITH robot specification