

**Comparing Synchrony Detection Algorithms  
For Robotic Self-Other Discrimination**

A THESIS

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL  
OF THE UNIVERSITY OF MINNESOTA

BY

Sampanna Salunke

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
MASTER OF SCIENCE

August 2005

UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of master's thesis by

**Sampanna Salunke**

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by the final

examining committee have been made.

**Dr. Christopher Prince**

---

Name of Faculty Adviser

---

Signature of Faculty Adviser

---

Date

GRADUATE SCHOOL

# Acknowledgements

I would like to take this opportunity to thank my advisor, Dr. Chris Prince for his guidance throughout this thesis, and indeed my entire graduate career. I gained a lot of valuable knowledge in the field of Developmental Robotics with his assistance.

Many thanks to Nathan Helder, Eric Mislivec and Anoop Reddy for helping me at various points in this thesis.

I also owe my heartfelt appreciation and thanks to Dr. Ted Pedersen and Dr. Fernando Rios-Gutiérrez for being a part of my thesis committee, and for helpful suggestions along the way.

I would like to acknowledge Jim Luttinen, Lori Lucia and Linda Meek for their help with infrastructural issues.

Finally, I would like to thank all my friends for their support and encouragement throughout the thesis.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Literature Review . . . . .	4
1.1.1	Developmental Robotics . . . . .	4
1.1.2	Synchrony & Contingency Detection Algorithms . . . . .	7
1.2	The Importance of Self-Other Discrimination to Robotics . . . . .	17
1.3	Algorithm Selection . . . . .	18
1.4	Defining Synchrony . . . . .	20
1.5	Thesis Outline . . . . .	23
<b>2</b>	<b>Self-Other Discrimination Using Hershey-Movellan</b>	<b>24</b>
2.1	Results . . . . .	29
2.2	Discussion . . . . .	31

<b>3</b>	<b>Self-Other Discrimination Using Optic Flow</b>	<b>38</b>
3.1	Physical World Simulation . . . . .	39
3.2	Optical Sensor . . . . .	41
3.3	The Need for Learning . . . . .	43
3.3.1	Structure of neural network . . . . .	44
3.3.2	Generation of training data . . . . .	46
3.3.3	Training and testing the network . . . . .	50
3.3.4	Discussion . . . . .	54
<b>4</b>	<b>Discussion</b>	<b>56</b>
4.1	Comparison . . . . .	57
4.1.1	Self moving, other stationary . . . . .	57
4.1.2	Self stationary, other moving . . . . .	58
4.1.3	Self moving, other moving . . . . .	59
4.1.4	Self stationary, other stationary . . . . .	59
4.2	Conclusion . . . . .	60
4.3	Limitations of the Fitzpatrick method . . . . .	60
4.4	Future Work . . . . .	62

A Tables	67
Bibliography	69

# List of Figures

1.1	Centroid Computation In An Implementation Of Hershey-Movellan (2000)	9
2.1	Experimental Setup - Modified Helder Implementation	27
2.2	Self And Other Moving, Higher Synchrony On The Side Of Self (Left)	34
2.3	Self Moving, Other Stationary	35
2.4	Other Moving, Self Stationary	36
2.5	Self Moving, Other Stationary - Effect Of Noise	37
3.1	Experimental Setup - Modified Fitzpatrick method	40
3.2	Structure Of Neural Network	44
3.3	Starting Positions For Automated Data Collection	46
3.4	Sample Optic Flow Vectors, Self Moving Downwards	47
3.5	Invisible Boundary Between Self And Other	48

3.6	Error Range Over Different Parameters . . . . .	50
3.7	Final Network Architecture . . . . .	52
3.8	Modified Classification Algorithm . . . . .	55

# List of Tables

1.1	Summary Of The Three Synchrony Detection Algorithms . . . . .	20
2.1	Results For Extended Helder Implementation . . . . .	30
2.2	Results With Thresholding Applied On The Output Mixelgram . . . . .	31
2.3	Results With Gaussian and Sobel Filtering Applied On The Output Mixelgram	31
3.1	Sample Training Data . . . . .	45
3.2	Distribution Of Training, Validation And Testing Datasets For Cases 1 & 2 .	49
3.3	Results Over Best Performing Parameter Set . . . . .	51
3.4	Breakdown Of Best Results For Cases 1 & 2 . . . . .	51
3.5	Results Distribution, Self And Other Moving . . . . .	53
3.6	2x2 Chi-square, Unrelated Vectors . . . . .	54
4.1	Summary Of Results . . . . .	58

A.1 Training Over Different Parameters . . . . .	68
--	----

# Abstract

Robot learning techniques that are based on infant psychological development have lately been gaining popularity. Since infants learn, in part, by detecting synchrony between their sensory and motor systems, we hypothesize that synchrony detection can be a useful basis for improving the adaptive qualities of robots. The goal of this thesis was to compare two time-based synchrony detection algorithms with respect to their ability to perform visual self-other discrimination. Self-other discrimination is the ability to distinguish between any effect caused by one's self from any effect caused by some other entity in the world. The two algorithms chosen for comparison were Hershey and Movellan (2000) and Fitzpatrick and Metta (2003). The Hershey and Movellan (2000) algorithm had already been implemented in our laboratory (Mislivec, 2004 and Helder 2003) and was extended in this thesis to make it capable of self-other discrimination. The Fitzpatrick and Metta (2003) method was implemented by us, with some modifications to suit our requirements. The Fitzpatrick and Metta method was able to discriminate between visual motion of self and visual motion of other with an accuracy of 74.34% as compared to 56.67% for Hershey and Movellan (2000). These results are for the most demanding case of self-other

discrimination, i.e., when both the self and the other are simultaneously in motion. With a trivial further modification, the method based on Fitzpatrick and Metta (2003) is at least as good as the Hershey and Movellan (2000) method in all cases considered. Thus we conclude that the Fitzpatrick and Metta (2003) method performs better under our test conditions than the Hershey and Movellan (2000) algorithm at our task of self-other discrimination. We also present suggestions for future research, including possible improvements to the performance of the two methods.

# Chapter 1

## Introduction

Researchers today are looking for novel methods for advancing robotics and constructing robots that are more independent and adaptable. One potential methodology is based on infant psychological development. Researchers are attempting to structure robotic learning based on the way infants learn new concepts and capabilities as they grow older.

From an early age infants learn, in part, by detecting synchrony between their sensory and motor systems and across different sensory modalities (e.g., Gergely & Watson, 1999). For example, infants learn object names more rapidly if the name is uttered while the object is moved in their view (Gogate & Bahrick, 1998). The mechanisms that enable infants to detect synchrony relations are one basis upon which we may potentially build robots that are more independent and adaptable. “Sophisticated behaviors can result from the application of simple mechanisms” (Prince, Helder, Mislivec, Ang, Lim & Hollich, 2003)

and hence we propose to model these synchrony detection mechanisms, in order to set a basis for constructing robotic systems along the lines of infant psychological development.

Following the reasoning that infants learn, in part, by synchrony detection, we propose to explore a range of algorithms that can detect perceptual-level synchrony. There are currently a number of algorithms that attempt to detect synchrony relations existing between various sensory and/or motor systems. Presumably, one algorithm will not be suitable for all synchrony detection tasks. Also, exploring various algorithms should enable us to establish a baseline for comparing the algorithms, and help us better understand the individual algorithms. To this end in this thesis we will compare two synchrony detection algorithms. The first algorithm we will consider (Hershey & Movellan, 2000) has been previously implemented in our lab in the context of detecting audio and visual synchrony in data obtained from digital video (MPEG) files (Mislivec, 2004), and also in the context of detecting animation-command and visual synchrony in data from a shape being animated on a computer screen and a video camera (Helder, 2003). Additionally, audio-visual synchrony detection in infants has been compared to a model of synchrony detection based on the Hershey and Movellan (2000) algorithm. Some good results and some differences from infant learning and behavior were observed from the model (Prince, Hollich, Helder, Mislivec, Reddy, Salunke & Memon, 2004). Also see Vuppla (2004) for further description and evaluation of the Mislivec (2004) and Helder (2003) programs.

The Hershey and Movellan (2000) algorithm (*HM* hereafter) enables detection of synchrony between a channel of visual sensory information and a second channel of

information. This second channel can be audio, or some other kind of data varying with time (e.g., an abstract description of a shape being animated; Helder, 2003). One candidate for the second algorithm we will use in this thesis as a comparison is a sensorimotor contingency detection technique that has been implemented recently in a robotic system by Fitzpatrick (2003; see also Metta & Fitzpatrick, 2003). Contingency detection deals with identifying when some events or stimuli are reliably related. Fitzpatrick used this technique in a robotic system in order to detect the visual sensory effects of moving the robot's own effector, and also contacting objects with the robotic effector. Another candidate for comparison is FaceSync (Slaney & Covell, 2001), an algorithm which quantitatively measures the synchrony between audio and visual streams.

In this thesis, our goal is to compare two synchrony detection algorithms. We will carry out this comparison in the framework of a standard task. The aim of the standard task is *self-other* discrimination. Self-other discrimination is the ability to distinguish self from the universe, e.g., a moving arm should be identified as the self's arm or someone else's. Gergely and Watson (1999) suggest that infants perform self-other discrimination by detecting synchrony relationships across sensory modalities and motor systems. The self-other task for this thesis will be as follows. There will be a robot arm which is capable of movement, and the movement of the arm and any objects that are not controlled by the robot will be observed by the robot's senses. Since we do not have a robotic system in our laboratory, we will simulate the movement and this simulated motion will be captured visually in a digital format. The corresponding commands sent to move the robot arm

(*agent* hereafter) will also be recorded. This will give two separate streams of information: the commands being sent to move the object (i.e., the arm), and the visual image of the moving object. The motion of the arm and the image of the moving arm are synchronized in time. The aim is to determine the relation between the motion of the arm and the commands sent to it, i.e., to develop some mechanism for discriminating between self and the world. Any movement in the world which is not caused by the robot itself should be ignored or possibly classified as “other.”

In the remainder of the Introduction we first present a literature review providing background information, and then briefly layout the contents of the body of this thesis and our initial expectations regarding the comparison of the synchrony detection algorithms. We also present the factors we used to select the algorithms for comparison, and since we are interested in synchrony detection, a brief description of synchrony as defined in fields including physics, computer science and developmental psychology.

## **1.1 Literature Review**

### **1.1.1 Developmental Robotics**

Developmental robotics is a new field which aims to combine the areas of robotics and developmental sciences such as psychology or neuroscience (Lungarella, Metta, Pfeifer & Sandini, 2003. See also: Prince, Helder & Hollich, 2005; Weng, McClelland, Pentland,

Sporns, Stockman, Sur & Thelen, 2001; Zlatev & Balkenius, 2001). The main goals that encourage the growth of this field are:

- To construct independent, adaptable and sociable robots.
- To investigate and test theories forwarded by neuroscientists and psychologists.

Lungarella, Metta, Pfeifer and Sandini (2003) present an overview of ongoing research in the field of developmental robotics. In recent years scholars of different disciplines have generally agreed that the brain, the body and the environment are not independent entities but are in fact reciprocally coupled (Lungarella, Metta, Pfeifer & Sandini, 2003). Thus it is currently understood that cognitive processes arise from having a body which can perform physical actions and from the interaction of this body with the world (See also Thelen, 2000). This view exhibits a marked change from more traditional views of cognitive science, which see the body as an output device that merely executes the commands generated by the mind. These commands were thought to originate from the mind in response to environmental conditions. In this traditional cognitive science view, it is assumed that there exists an internal representation of the world inside the mind.

Currently there exist synthetic approaches that seek to understand the process of cognitive development by building physical models (i.e., robots) on which cognitive models can be tested. These physical models are often built in an incremental manner, starting with a smaller, simplified robot and then gradually adding to the capabilities of the robot. The cognitive models tested on these robots are also built incrementally, starting with a

simpler model and building up to more complex models. Physical development and learning are related and there are three leading views about the nature of the interdependency between physical development and learning. The first view says that learning uses development, and is termed *unidirectional*. This view is unidirectional in the sense that learning, it is hypothesized, cannot occur unless the developing child has a certain level of physical development. The second view is *bidirectional* and says that learning and physical development are mutually coupled. The physical development encourages learning, and learning in turn helps in advancing physical development. For example, as infants' physical object exploration skills increase, their ability to cognitively separate objects based on the physical characteristics of the object seems to increase (Needham, 2000). These two views suggest that development occurs incrementally, and the direction it takes is always forward. The third view, presented by Thelen and Smith (1994) suggests that the boundaries between development and learning should be erased, while considering the 'dynamics' at all levels of organization. They give evidence that suggests that development does not necessarily proceed in an incremental manner. There are instances where development is not linear, and may even regress. For example, infants learning to walk do not seem to go clearly through McGraw's "seven phases of erect locomotion" (McGraw, 1945). The stepping motions performed by infants who are held erect seem to disappear at about 2 months of age, and reappear after about 6 months (e.g., see Thelen & Fisher, 1982).

### 1.1.2 Synchrony & Contingency Detection Algorithms

Psychological and physiological observations lead us to believe that localization of sound sources occurs in part based on the synchrony of the audio and the visual signals. This is demonstrated in the case of a television or a ventriloquist's dummy. When watching a movie on the television, speech seems to originate from the lips of the speaking person when the audio and visual information are correctly synchronized even though the sound source is actually spatially dislocated from the visual motion. This effect, whereby the sound is thought to be located at the apparent dynamic visual source is known as the *ventriloquism* effect. The effect disappears when the visual signals are not synchronized with the audio signals. For example, the ventriloquism effect disappears when there is a lag between the movement of the speaker's lips on the screen and the words being uttered by the person. Hershey and Movellan (2000) present an algorithm to evaluate any synchrony that exists between audio and visual data and apply this algorithm to the problem of sound-source localization. They developed a system that searches the visual data for regions or segments that correlate highly in time with the audio signals. These segments are then labeled as possible synchronized audio sources.

Synchrony here is defined as “the degree of mutual information between the audio signals and spatially localized video signals” (p. 814; Hershey & Movellan, 2000). A sequence of vectors of audio and visual data is considered, sampled at different consecutive times and for visual data, at different spatial coordinates. The idea here is to provide a collection of numbers that describe the synchrony between the audio and the visual signals

at any time. Mutual information (interpreted as audio-visual synchrony) is calculated by computing estimates of a covariance matrix where each vector is taken as an independent sample from a joint multivariate Gaussian process. This is given by Equation 1.1 (from Hershey & Movellan, 2000).

$$I(A(t_k); V(x, y, t_k)) = \frac{1}{2} \log_2 \frac{|\sum_A(t_k)| |\sum_V(x, y, t_k)|}{|\sum_{A,V}(x, y, t_k)|} \quad (1.1)$$

where

$A(t_k)$  is the audio vector over a time window of length  $S$ ,

$V(x, y, t_k)$  is the visual vector over the same time window of length  $S$  for the pixel located at position  $(x, y)$  in each of the visual frames,

$I(A(t_k), V(x, y, t_k))$  is the mutual information between the audio and the visual information.

A prototype of the system was created using a Logitech QuickCam on Linux and then ported to Microsoft Windows NT (Hershey & Movellan, 2000). The challenges faced during implementation were processing time-synchronized audio and visual information on a serial machine in real-time. With separate threads handling the audio and the visual data the two data streams had to be buffered, time-stamped and synchronized before they could be processed.

To create a performance baseline for this synchrony detection method the authors considered the simplest condition first: a single audio and visual signal per location. It was

observed that averaging over many small windows produced more information than a single large window. The mutual information measure was applied per pixel and the result was interpreted as a dynamic topographic map of the synchrony present in the audio-visual information. The position of the person talking was estimated by computing a centroid where each point was weighted by the mutual information between that pixel and the audio signal. Figure 1.1 shows a centroid computation over the mutual information output in the Mislivec (2004) implementation which uses HM.



Figure 1.1: Centroid Computation In An Implementation Of Hershey-Movellan (2000)

The HM paper presents exploratory work in localizing sound sources in visual images by marking segments of the image that seem to correlate with the audio signal. The results presented are encouraging, and the paper concludes by saying that more work along the lines of investigating more sophisticated methods for processing the audio and visual

signals needs to be done before practical applications are developed.

We chose HM as one of the algorithms for comparison for programmatic research considerations in our laboratory. That is, we wanted to explore the degree to which the HM algorithm could be used with different applications. For example, previously the Helder (2003) implementation showed the possibility of detecting synchrony between visual images and a stream of commands and the Mislivec (2004) implementation complemented the original HM application by detecting synchrony between audio and visual streams. The first algorithm we consider for possible comparison with HM has been developed by Slaney and Covell (2001). These authors present an algorithm that measures the degree of synchronization between the dynamic visual image of a person and the audio of the person talking. This algorithm generates an audio-visual synchronization error signal in a computationally inexpensive manner. It combines all the pixel information available to quantitatively measure the audio-visual synchronization, unlike HM which processes an audio signal and individual pixels of the visual signal, giving qualitative mutual information between the audio signal and the visual signal.

FaceSync is based on two algorithms: Eigenpoints (Covell & Bregler, 1996) and a multilinear facial synthesizer (Yehia, Rubin & Vatikiotis-Bateson, 1998). Eigenpoints finds a linear mapping between the brightness of a visual signal and the position of fiduciary points on a face. The multilinear facial synthesizer shows a connection between specific modes of speech with the location of fiduciary points on the face. Fiduciary points are those points that are used as reference points on the face in the marking of a measuring

scale in an optical instrument. FaceSync combines the ideas from both Eigenpoints and the facial synthesizer to directly connect brightness to audio without the fiduciary points. It describes a linear transformation between the brightness of the visual signal and the audio.

FaceSync measures the audio-visual synchrony in two steps: training a canonical correlation model and then evaluating how well the model fits the data. As a prerequisite of the algorithm, face recognition software is used to find faces and align them to a sample image. A neural-network face-detection algorithm is used to find faces in the images, and the audio signal is described as Mel-Frequency Cepstral Coefficients (MFCC's; Rabiner & Juang, 1993), which throws away pitch information. In the first step, canonical correlation finds a linear mapping that maximizes the cross-correlation between the aligned face image and the audio signal. This mapping is used to rotate an audio signal and a new aligned face, given new audio and visual data, and then their Pearson's correlation is evaluated over time.

Slaney and Covell's evaluation of the performance of the FaceSync algorithm measured the sensitivity of the algorithm to small temporal shifts between the audio and visual signals of someone talking. The tests also measured the effect of coarticulation. Coarticulation is the mechanism wherein lip, jaw and tongue movements are coordinated by a human speaker to ensure that speech is produced smoothly. MFCC's, Linear predictive coding (Rabiner & Juang, 1993) and Line spectral frequencies (Sugamara & Itakura, 1986) when used to describe the audio signal produced sharp correlations, but speech power and spectrograms did not produce any interesting correlation.

Another algorithm that falls in the category of synchrony detection algorithms has been developed by Fitzpatrick and Metta (2003), who present a robotic method for acquiring visual experience through exploratory physical motion. The work was implemented on Cog, an upper torso humanoid robot, capable of vision by means of two cameras per eye mounted on a head. Cog also has two arms, one of which was used in this experiment to move objects and interact with the environment. The authors attempt to exploit advantages provided by this interaction of the robot with its environment. They worked on the idea that visual segmentation of an object from its background is possible by pushing the object gently with a robotic arm; they track the movement of the robot's arm and look for motion (other than the motion of the arm) which seems to be physically adjacent to the arm. For this purpose, the first task is to localize the arm itself.

The sequence of steps involved in segmenting the object from the background is:

1. Localization of the arm

- 1.1. The robot arm is localized by looking for correlations between the motor command sent to the arm to move it and the visual image of the moving arm. The method used for localizing the arm was optic flow. In an image, each pixel corresponds to the intensity of an object in space projected onto the image plane. When objects move, their corresponding projections also move. Optic flow is a vector field that shows the magnitude and direction of the changes in intensity from one image to another. To achieve the initial arm localization, the arm was commanded to

rapidly switch directions and this caused the optic flow at the times of the direction switch to change in numeric sign. Thus the authors obtained a tight correlation in time between the motor commands and the visually observed arm movement.

Fitzpatrick and Metta (2003) used this method to locate the robot arm in the camera image. Here the arm is assumed to be the only moving object in the scene.

1.2. After the initial arm localization, a function is trained which can estimate the next arm location based on the command to be executed. It is impractical to attempt localization of the arm after every movement of the arm by using optic flow (i.e., method 1.1), hence this prediction is necessary. Also, any movement in the scene when the arm is stationary can now be ignored as a distractor.

2. When the robot knows about its arm (i.e., it can predict the next location of the arm based on the current location and the command to be executed next), it can then use the arm to explore the local environment. If the arm encounters an object during its exploration, the object may move when it is touched. The arm can then be removed from the image using its predicted position. Optic flow past a threshold distance from the end-effector is ignored as a distractor (i.e., noise). The hypothesis here is that physical distance from the arm in the image observed by the camera represents motion unrelated to the arm.
3. The image obtained by the method of frame subtraction (i.e. by subtracting the arm using its predicted position from the observed camera image) may not give a good segmentation of the object. For example, if the moving object is a box, then the only

pixels identified as part of the object may be two horizontal lines.

3.1. In order to correctly segment the object, it is necessary to determine which of the pixels are part of the object (considered a part of the foreground) and which are part of the world (considered the background). A technique called background modeling was used to estimate the background and foreground parts of a scene. See Fitzpatrick (pp. 39–42, 2003) for details of this technique. In this technique the idea is to use pixel variation and not true motion and categorize each pixel as either a foreground pixel or a background pixel. The foreground is the object they want to segment and the background is everything else in the image. Information about which category a pixel belongs to is sparse and can be represented as probabilities that a pixel belongs to either the foreground or the background.

3.2. They look first at a simpler part of the segmentation problem where they know foreground/background information for some pixels in the image (e.g., the pixels corresponding to the arm of the robot). Following this, their method is to assign every pixel in the image to the foreground or the background using the information available to the robot. One approach to this assignment is to create a cost function. This cost function can be used to evaluate different possible segmentations and choose the one with the minimum cost. They use a family of maximum-flow/minimum-cut algorithms that can provide a good approximate solution to this problem (Boykov & Kolmogorov, 2001) if a constrained cost function is used. To apply these algorithms they translate the problem to a graph. Every pixel

maps to a vertex in the graph, and is connected to the vertices representing the neighboring pixels by an edge. The minimum-cut algorithms split the graph into two disjoint graphs, with the foreground vertex in one and the background vertex in the other such that the total cost of the edges being removed in making this split is minimized. Costs are then assigned to the edges such that a minimum cut of the graph corresponds to a sensible segmentation.

Working on the same robot (Cog), Arsenio and Fitzpatrick (2003) presented a mechanism for a robot to perceive moving objects. The authors base their method on the often rhythmic nature of the movement of an object and the sound generated by that motion (e.g., a hammer striking a surface). The authors use visual and acoustic information to detect simple repeated events. These two modalities (i.e., visual and acoustic) provide complementary information; sound waves provide robustness to occlusion while vision provides more spatial structure than sound.

To detect periodicity, the authors initially used the Short-Time Fourier Transform (STFT), which maps a signal into a two-dimensional function of time and frequency. However, STFT did not work very well in their experiments to detect periodicity in acoustic signals, which vary considerably in amplitude between periods. This led the authors to create a different method for periodicity detection, which they applied to both the acoustic and the visual signal. A histogram of durations between successive instances of particular signal values was constructed. The most common duration was then taken as the period of the signal. Matching sound and vision was done by the authors based on the

following assumption: the sound produced by the object when it is moving is highly correlated with the motion of the object and the visual image of the moving object. If the periods of the acoustic and the visual signals are within a tolerance of 60ms then they are grouped/bound together.

Three cases of cross-modal binding were investigated by the authors. The first case (matching with visual distraction) dealt with multiple moving objects that were visible with only one repeating sound. In the second case (matching with acoustic distraction) two moving objects (one in the robot's view and the other off-stage) and the corresponding generated sounds were considered. The final case (matching multiple sources) considered two moving objects in the robot's view, both generating sounds. The results of the various experiments showed that it is possible to bind rhythmic motion with sound and the authors obtained particularly strong binding in the case of a hammer striking a surface.

This approach can be also be used in an object segmentation task by finding pixels and frequency bands (respectively) that oscillate together, thereby distinguishing objects from their backgrounds, both visually and acoustically.

## 1.2 The Importance of Self-Other Discrimination to Robotics

The idea of using the metaphor of infant development in robotics is new and has primarily come about in research in the last six years (e.g., Metta, Sandini & Konczak, 1999). The motivation for using the infant development metaphor in robotics is both in the potential for greater behavioral adaptability by robots, and the use of robots in psychological research as models. With somewhat of a longer history, robotics has started to emphasize a closer, dynamical, interaction with the environment (e.g., for a collection of papers see Brooks, 1999). Self-other discrimination could potentially lead to perceptual self-awareness by the robot, so that it is capable of interaction with a dynamic environment and able to understand its effect on the environment. We hypothesize that algorithms that enable the development of self-other discrimination skills can enable a robot to exhibit greater behavioral adaptability. Instead of programming a robot to perform a particular task, we could program it to develop. Self-other discrimination could lead to perceptual self-awareness. If a robot is perceptually self-aware, it could learn new tasks that build on this self-awareness more quickly, in a boot-strapped manner.

In other contexts, e.g., without emphasis on the metaphor of child development, some research has made contact with self-other discrimination. Talukder and Matthies (2004) present a method for real-time recognition of moving objects. The target of their research is autonomous vehicles operating around humans and other moving objects. They use optic

flow analysis and egomotion estimates to identify “other” moving objects and distinguish them from the robot’s self. Egomotion is defined as the displacement of the observer in the environment. Kuffner, Nishiwaki, Kagami, Kuniyoshi, Inaba and Inoue (2002) presented a method for humanoid robots to detect self-collision. The achievement of such a goal indicates that the robot is learning some degree of perceptual self-awareness. Morency and Gupta (2003) describe a technique to compute egomotion from stereo images. They use depth perception information available from the stereo camera for this purpose. Again, there seems to be some mechanism to discriminate self from other to compute egomotion. The emphasis of these research projects is not on development, however, in each case the robotic system appears to learn some mechanism to perform self-other discrimination.

### **1.3 Algorithm Selection**

The long term goal of this research is to compare synchrony detection methods, in order to establish a basis for constructing robotic systems along the lines of infant psychological development. As one of the algorithms in our comparison we will use HM (Hershey & Movellan, 2000) because of programmatic research considerations, as stated earlier. Our two possible other algorithms to be used for comparison are FaceSync and Fitzpatrick’s method. The comparison will be performed in the framework of a standard self-other discrimination task. The standard task is as follows: The movement of the agent’s arm will be observed by the agent’s visual senses. Any objects which come in contact with the

agent’s arm will also move, and this movement will also be observed. The aim is to determine which movement in the world has been caused by the agent’s commands to the arm. This can be interpreted a perception of “self” and any other movement in the world can be classified as “other”.

Table 1.1 briefly summarizes the salient features of the three main algorithms reviewed in the previous section. The row labeled *Task* describes the task the algorithm was originally used with, and the *Technique* row shows the methodology used in the algorithm. The *Equipment* row lists the hardware required for an implementation of the algorithm. The row labeled *Generic* relates to the type of input streams to the algorithm. That is, can the types of inputs to the algorithm be changed? Can an audio stream be replaced by say, a stream of commands, or is the algorithm bound to using specific types of input streams? Between FaceSync and Fitzpatrick’s method, Fitzpatrick’s method provides a better candidate to perform a self-other discrimination task. FaceSync determines the synchrony between the lip movement in a person’s face and the audio signal. It is unclear if self-other discrimination can be applied as a performance measure can be applied to FaceSync. Possibly, the types of data channels used as input for FaceSync do not have to be audio and visual but could instead be commands and visual. Specific experimentation would be required to evaluate this possibility. However, Fitzpatrick’s method already attempts to distinguish between motion caused by self and unrelated motion in the world. Given this rationale, we choose Fitzpatrick’s method as the second algorithm in our comparison.

	<i>HM</i>	<i>FaceSync</i>	<i>Fitzpatrick</i>
<i>Task</i>	Sound source localization	Sound source localization	Object segmentation
<i>Technique</i>	Covariance matrix	Pearson's correlation	Optical flow analysis
<i>Equipment</i>	Camera	Camera	Robot
<i>Generic</i>	Partially	Possibly	Task specific
<i>Segmentation</i>	Yes	No	Yes

Table 1.1: Summary Of The Three Synchrony Detection Algorithms

## 1.4 Defining Synchrony

Since the topic of the present research is synchrony detection, it seems useful to consider various interpretations of the word ‘synchrony’. The Webster’s Revised Unabridged Dictionary (1996) defines synchrony as the “concurrence of events in time”. In infant psychological development Gergely and Watson (1999) talk about three different types of synchrony:

- **Temporal Synchrony:** When changes in two signals occur simultaneously in time, they may be causally linked. For example, moving one’s arm results in some visual stimulus. The command sent to move the arm is one stream of information and the visual images observed by the eyes is the other source of information. The commands

and the visual images are temporally synchronized. Temporal synchrony exists when the same kind of information occurs repeatedly at differing points in time.

- **Sensory Synchrony:** Sensory synchrony deals with the relationship between the energy put into any act and the amount of sensory consequence obtained. For example, tapping a hammer on a surface produces a mild sound, while striking the hammer harder produces a much louder sound.
- **Spatial Synchrony:** Spatial synchrony refers to the positions of objects. According to Gergely and Watson (1999), instead of relying on synchrony across time or amount of energy, the location of different objects may help us to remember some event. For example, we may remember a speaker and his speech by remembering the locations of different objects that surrounded him. But, we find the idea of spatial synchrony presented by Gergely and Watson (1999) hard to conceptualize. Our interpretation of Gergely and Watson's (1999) idea of spatial synchrony is that this property occurs when the same kind of positional information occurs repeatedly at different points in space. For example, a gas station is frequently found physically close to a freeway exit.

In physics, the repeating and periodic disturbance which moves through a medium from one location to another is referred to as a wave. The frequency of a wave refers to how often the particles of the medium vibrate when a wave passes through the medium. The period of a wave is the time for a particle of the medium to make one complete

vibrational cycle and the phase is the fraction of a complete cycle elapsed as measured from a specific reference point and often expressed as an angle. Signals are said to be synchronous when they are in phase, and this applies when the amplitude and frequency of the two waves is identical. When two waves meet while traveling in the same medium, interference takes place. Interference is of two types:

- **Constructive Interference:** Constructive interference is a type of interference which occurs at any location along the medium where the two interfering waves have a displacement in the same direction.
- **Destructive Interference:** Destructive interference is a type of interference which occurs at any location along the medium where the two interfering waves have a displacement in the opposite direction.

Interference distorts information available when two signals are in phase. Constructive interference between two signals increases the strength of the resulting signal. Thus more data is available for analysis when two signals are synchronized.

In telecommunication signaling within a network, synchronous signals are those that occur on the same clock. When signals are not issued on the same clock, the communication mechanism between networks may fail. Thus synchrony between signals or events is important across various disciplines, and synchronized signals strengthen the resulting signal.

## 1.5 Thesis Outline

In the remainder of this thesis, we will present an extension to the Helder (2003) implementation which uses HM, and evaluate the extended implementation with respect to self-other discrimination. We will also present our implementation of Fitzpatrick's method, and the evaluation for the same. We expect that both methods will be capable of some degree of self-other discrimination. Our question is: Which algorithm performs better at the task of self-other discrimination? We also expect to see that the results from HM are more difficult to quantify than the results from the implementation of Fitzpatrick's method, because the HM output is qualitative, not quantitative (See Figure 1.1).

## Chapter 2

# Self-Other Discrimination Using Hershey-Movellan

The goal of this thesis is to compare two synchrony detection methods for their ability to perform self-other discrimination. The two algorithms chosen for this purpose are HM (Hershey & Movellan, 2000) and Fitzpatrick's method (Fitzpatrick & Metta, 2003). To conduct this comparison, an implementation of the two methods is desired. HM has been previously implemented in our laboratory (Helder, 2003 and Mislivec, 2004). The Mislivec (2004) implementation does audio-visual synchrony detection, but because we are not utilizing audio inputs, this was not directly relevant to the present thesis. The Helder (2003) implementation performs synchrony detection between changing properties of a displayed shape and commands sent to change those shapes. The physical world simulation in the Helder (2003) implementation has an animation component controlled by the agent.

We will refer to this animation component as being controlled by the “self”. Because we needed the system to be capable of self-other discrimination, we extended the Helder (2003) implementation to include an “other” component, i.e., an animation component not controlled by the agent. The implementation of the Fitzpatrick method is described in more detail in Chapter 3.

The Helder (2003) implementation detects synchrony across two input streams using the HM algorithm<sup>1</sup>. The first stream is the sequence of commands sent to animate an object on the screen, and the second stream is the series of images captured by a webcam. Synchrony between the visual stream and the commands is computed in terms of a matrix of mixel values, also called a mixelgram. Each entry in the matrix is called a *mixel*, which is a name derived from the term *Mutual Information Pixel*. A mixel gives an estimate of synchrony between the corresponding pixel in the visual image and the stream of commands. The dimensions of the mixelgram are the same as the dimensions of the visual data, each pixel in the image has a corresponding entry in the mixelgram. The software has been implemented using Java and uses the Java Media Framework to read images from the webcam. For details, see Vuppla (2004). In this thesis the webcam used is a Creative NX Ultra (Creative Labs, California) working under Microsoft Windows XP.

---

<sup>1</sup>See: <http://www.cprince.com/PubRes/Detect>

The Helder (2003) implementation was extended to add an “other” component<sup>2</sup>. The “other” component was introduced in the system by adding an animation component controlled by a random process. The “self” and the “other” components in the extended program are animation objects. The stream of commands sent to change the self animation is one stream of information to the HM algorithm, the images captured by a webcam and synchronized in time with the commands is the other. We used “forced” synchrony between the images and commands, i.e., each time either of the objects moved, an image was acquired from the webcam. The drawing of the objects and the image acquisition could not be implemented as two independent threads since the windowing component of an operating system is typically asynchronous and does not guarantee the time at which an image will be drawn on the screen. We followed a technique of drawing an image, waiting for an interval of time for the image to be drawn, and then acquiring an image from the webcam. The experimental setup for the modified Helder (2003) implementation is displayed in Figure 2.1. As seen from Figure 2.1 and Figure 3.1, the experimental setups for the HM and Fitzpatrick implementations are very similar. This was intentionally done to facilitate comparison.

The animations performed in the Helder implementation involved changes in shape, size, location and color. Since our implementation of the Fitzpatrick method tracks changes in the location of an object (by virtue of using optic flow), for the modified Helder implementation we restricted the animation to using changes in location of the “self” and

---

<sup>2</sup>See: <http://www.cprince.com/PubRes/SalunkeThesis05>



Figure 2.1: Experimental Setup - Modified Helder Implementation

the “other” objects. The window size used for synchrony computation in this thesis was 8 (i.e.,  $S = 8$ ). We also changed the Helder implementation to reduce the number of steps required for an animation, thus increasing the visual change observed on the screen per unit time. The camera viewed both the “self” and the “other” objects simultaneously. Synchrony was computed across the sequence of commands sent to the self object and the visual images captured by the webcam. Our hypothesis was that the movement of the self would be more synchronous to the commands being sent, and hence the synchrony value computed for the self would be higher than the synchrony value for the other. That is, with the self in the left half of the image and the other in the right half, if the self is moving, then the synchrony estimate on the left side of the mixelgram would be higher than the synchrony estimate on the right half of the mixelgram. This hypothesis stems

from preliminary observation as seen in Figure 2.2 which shows the mixelgram output observed when both self and other objects are moving, with the other being controlled by a random process. The test setup was under uniform lighting to avoid changes in the visual image due to changes in illumination.

Since the Helder implementation only allowed self motion, we also extended it to allow the following three cases:

1. Self moves, other stationary.
2. Self stationary, other moves.
3. Both self and other move.

Through the rest of the thesis we will refer to these as Case 1, 2 and 3 respectively. Figures 2.3 and 2.4 show mixelgram outputs for Cases 1 and 2 (Figure 2.2 shows Case 3).

In order to obtain a quantitative estimate of synchrony, the output of the synchrony detection algorithm, i.e., the mixelgram, was further processed using thresholding, Gaussian filtering and Sobel edge detection to remove noise (e.g., see Prince *et al.*, 2004). This filtered mixelgram was then divided into two halves, the left half was the mixelgram output related to self and the right half was the mixelgram output related to the other. The filtered mixel values in each half were summed to a single value. The summation technique was used because we have had some success with it when applied to audio-visual synchrony (Prince, *et al.*, 2004). We are now trying the same technique on command-visual

synchrony. This value was used to classify the output as related or not related. For example, in Case 1 (self moving, other stationary) we expect the synchrony value to be higher on the side of the self object. In this case the commands and the visual motion observed on the side of the self would be classified as related.

If the degree of synchrony on the left is greater, then we classify the motion as related otherwise we classify it as unrelated. One consequence of this classification rule is that equal degrees of synchrony on both sides is then biased towards an “unrelated” classification. This is useful when the degree of synchrony on the left and right halves are both equal to zero. With this means of classifying the output, there is a 50% possibility that, by chance, the algorithm will classify correctly.

## 2.1 Results

The extended Helder implementation was set to run for 300 steps in each of the three cases described above (i.e., 900 total steps). Table 2.1 lists the results for the three cases, with simple summation (no threshold or Gaussian filtering and Sobel edge detection) over the output mixelgram. Table 2.2 shows the results with thresholding applied over the output mixelgram. The threshold is calculated by computing the standard deviation of the mixel values for the mixelgram. Any mixel values below the threshold are set to zero. Table 2.3 lists the results with Gaussian filtering and Sobel edge detection applied to the output mixelgram. In each of the above three tables, the column labeled *P-value* is the p-value

<i>Case</i>	<i>Correct</i>	<i>Incorrect</i>	<i>% Correct</i>	<i>P-value</i>
<i>1 (self moving, other stationary)</i>	218	82	72.67	<0.0000001
<i>2 (self stationary, other moving)</i>	300	0	100	<0.0000001
<i>3 (self moving, other moving)</i>	153	147	51	0.386443*

\* Not significant

Table 2.1: Results For Extended Helder Implementation

computed by applying the cumulative binomial test to the results. The null hypothesis is that there is no correlation in the population observed, i.e., the discrimination results are due to chance. A p-value is a measure of how much evidence we have against the null hypothesis, if the p-value is less than 0.05, we can reject the null hypothesis with 95% confidence. The null hypothesis can be rejected for Case 1 (self moving, other stationary) and Case 2 (self stationary, other moving) in all three quantifications (Tables 2.1, 2.2 and 2.3, thus indicating that for Cases 1 and 2 the discrimination results observed are not due to chance. For Cases 1 and 2, the best performing quantification yielded accuracy results of 71.67% and 100% respectively (Table 2.3). However, for Case 3 (self moving, other moving), the null hypothesis can be rejected only in the quantification involving Gaussian and Sobel filtering. This quantification results in an accuracy of 56.67% for Case 3.

<i>Case</i>	<i>Correct</i>	<i>Incorrect</i>	<i>% Correct</i>	<i>P-value</i>
<i>1 (self moving, other stationary)</i>	215	85	71.67	<0.0000001
<i>2 (self stationary, other moving)</i>	300	0	100	<0.0000001
<i>3 (self moving, other moving)</i>	135	165	45	0.963344*

\* Not significant

Table 2.2: Results With Thresholding Applied On The Output Mixelgram

<i>Case</i>	<i>Correct</i>	<i>Incorrect</i>	<i>% Correct</i>	<i>P-value</i>
<i>1 (self moving, other stationary)</i>	215	85	71.67	<0.0000001
<i>2 (self stationary, other moving)</i>	300	0	100	<0.0000001
<i>3 (self moving, other moving)</i>	170	130	56.67	0.012091

Table 2.3: Results With Gaussian and Sobel Filtering Applied On The Output Mixelgram

## 2.2 Discussion

In this chapter we described and evaluated the extension to the Helder (2003) implementation of the Hershey and Movellan (2000) algorithm. The HM algorithm computes synchrony between two input streams. In our extension of the Helder (2003) implementation, the series of visual images formed one input stream and the series of commands formed the second input stream. The output was classified as “related” or “not

related” using a summation technique (Prince, *et al.*, 2004). In the best performing quantification, which used Gaussian filtering and Sobel edge detection, the method was able to discriminate between self and other with 71.67% accuracy for Case 1 (self moving, other stationary), 100% accuracy for Case 2 (self stationary, other moving) and 56.67% accuracy for Case 3 (self moving, other moving).

The HM algorithm provides an output which, for our purposes, must be analyzed quantitatively. The quantification analysis of the Helder implementation prior to this thesis was subjective, involving human coders (Vuppla, 2003). Our attempt to automatically analyze the output quantitatively is based on previous success with the method (Prince, *et al.*, & Memon, 2004), however the technique is susceptible to noise. Figure 2.5 shows a situation where the self object is the only moving object in the scene, but the noisy mixels on the right half sum to a higher value, resulting in an incorrect classification. One of the positive aspects of HM is that the inputs to the HM algorithm are fairly generic, so it is possible to easily substitute commands to move the object with commands to change the color, size or shape of the object. Changes in color, size and shape might provide better discrimination and help reduce noise.

Since the HM algorithm uses a particular window size ( $S = 8$  used here), the first  $S$  frames have no output. Also, because of the window, if the animation changes from say self moving to self stationary, the  $S$  computations following that change would continue to consider the moving self, even if this is no longer true. This would also apply if the camera itself were to move. Note that this is not an issue in the current implementation since the

camera is stationary, however, this could pose a problem if the technique were applied to a robot capable of motion. The robot could potentially remain  $S$  frames behind the world.

One of the reasons for getting poor performance in the Case 3 may be because the self and the other are moving at the comparable rates. That is, even if the directions are different, the changes at the level of pixels may be similar. If there is no significant difference between the number of pixels changing and the extent of the change, the algorithm may be unable to classify them properly.

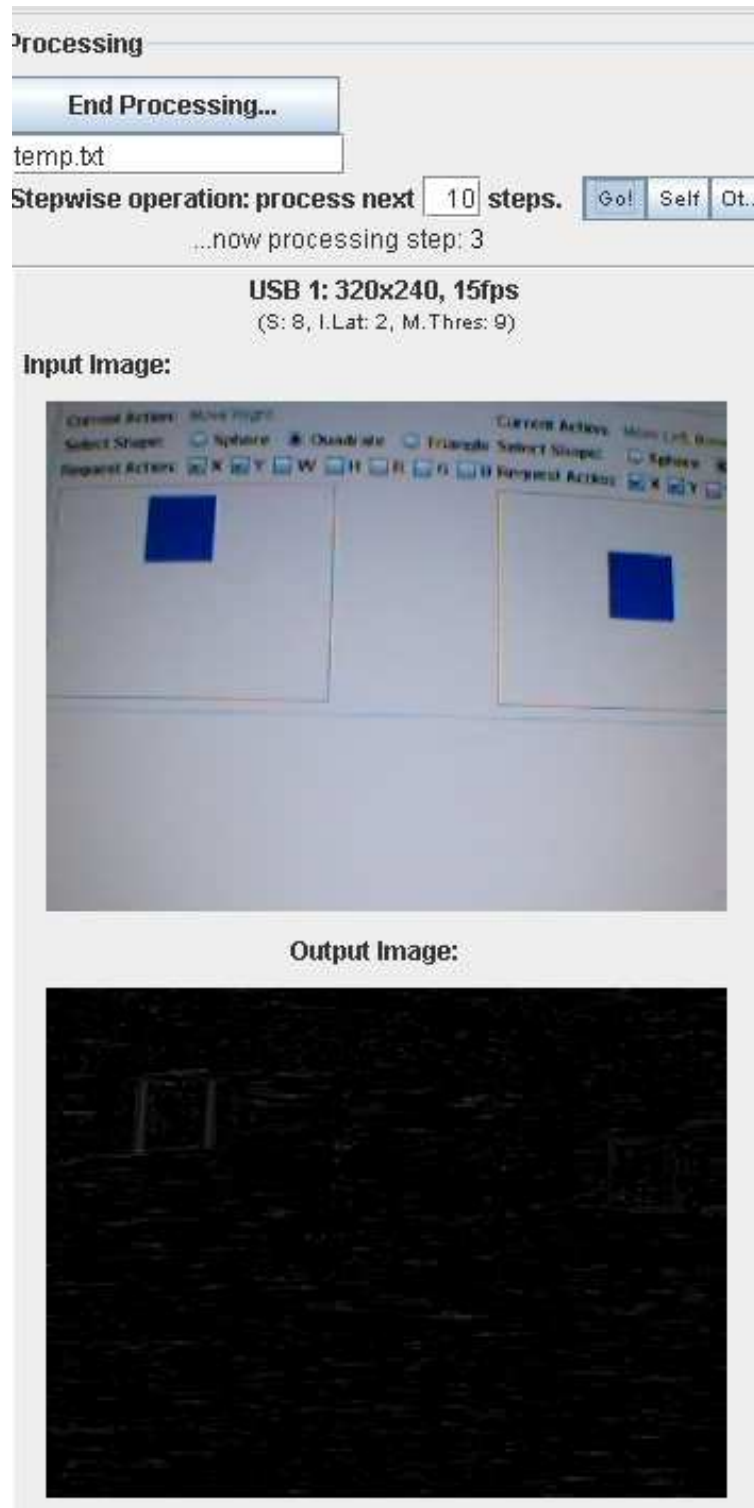


Figure 2.2: Self And Other Moving, Higher Synchrony On The Side Of Self (Left)

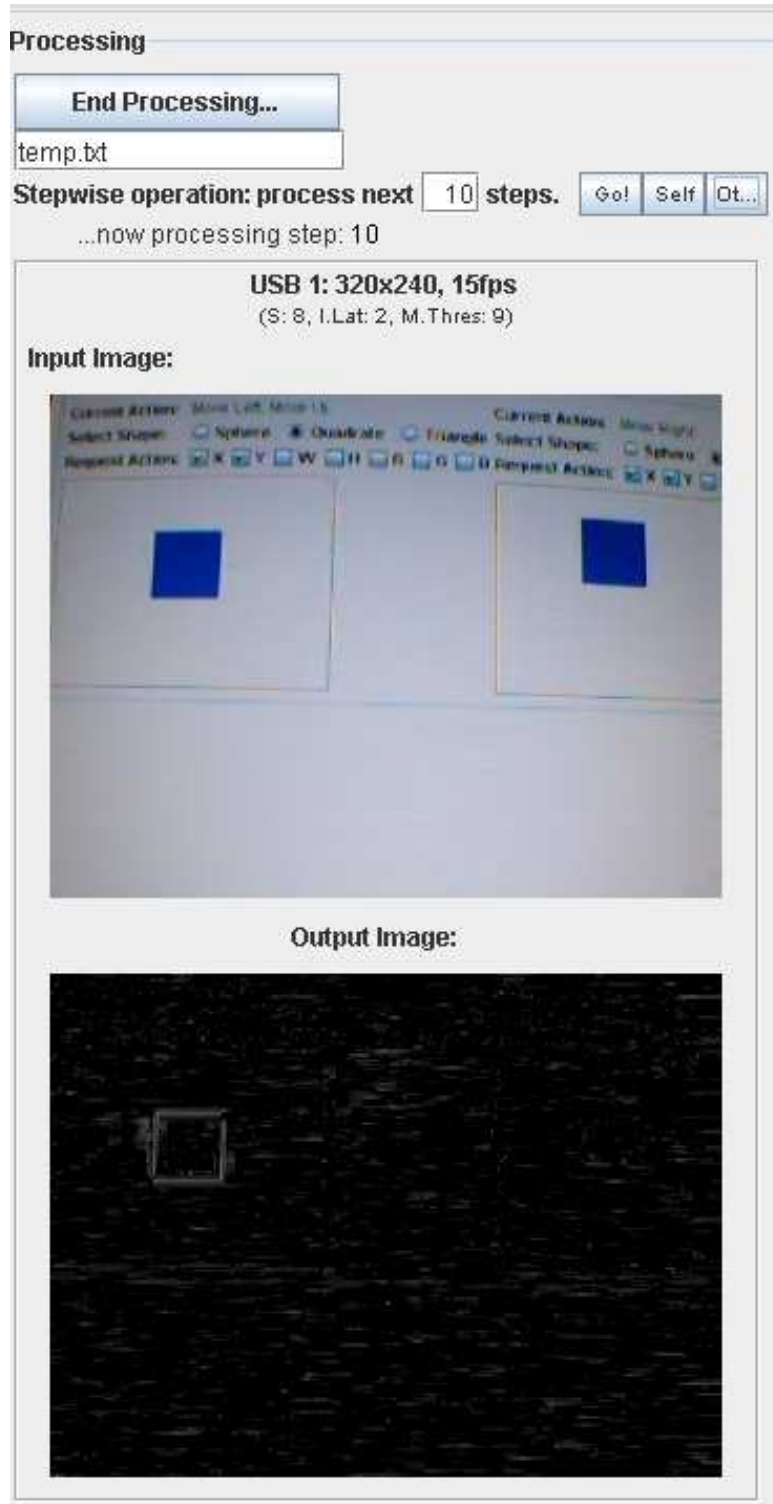


Figure 2.3: Self Moving, Other Stationary

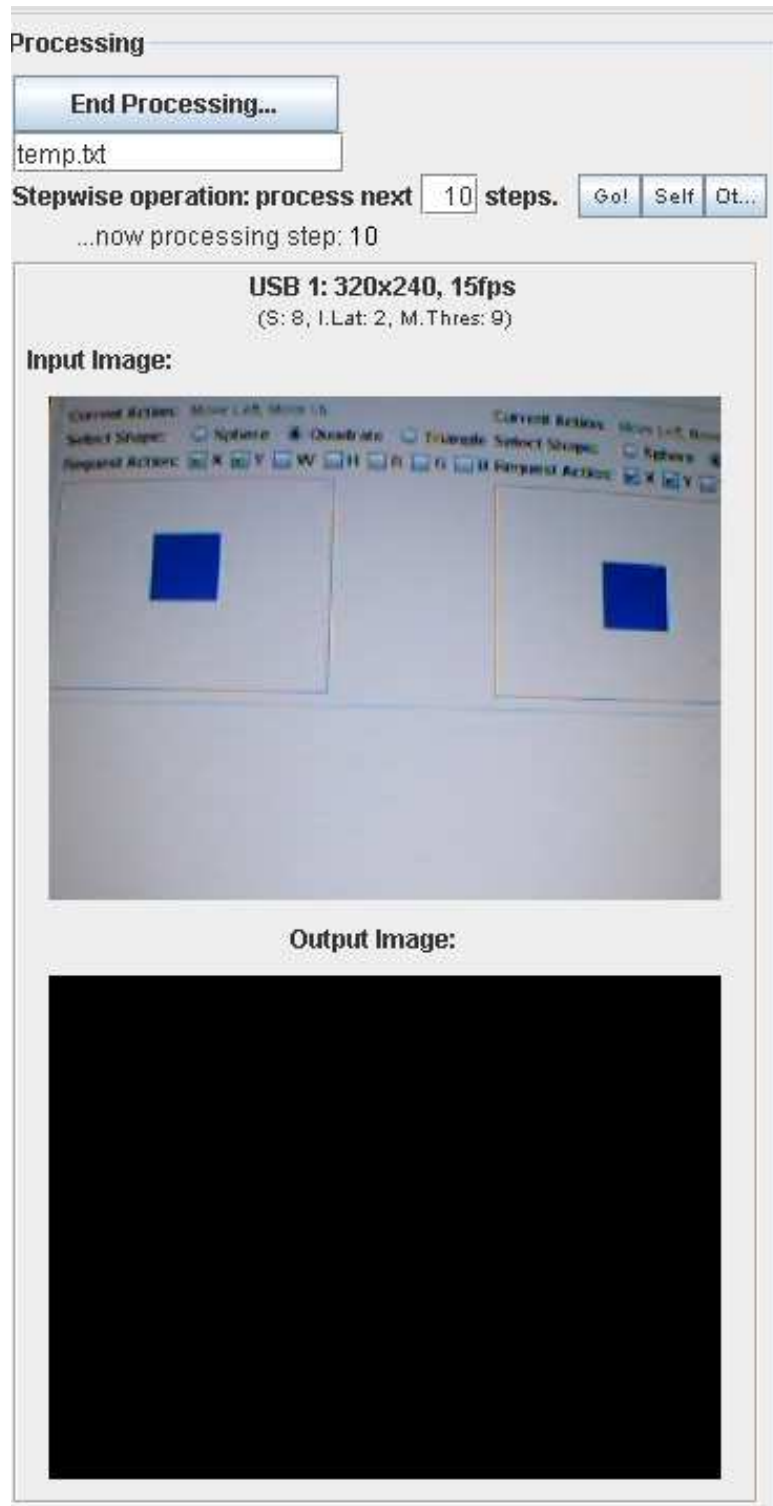


Figure 2.4: Other Moving, Self Stationary

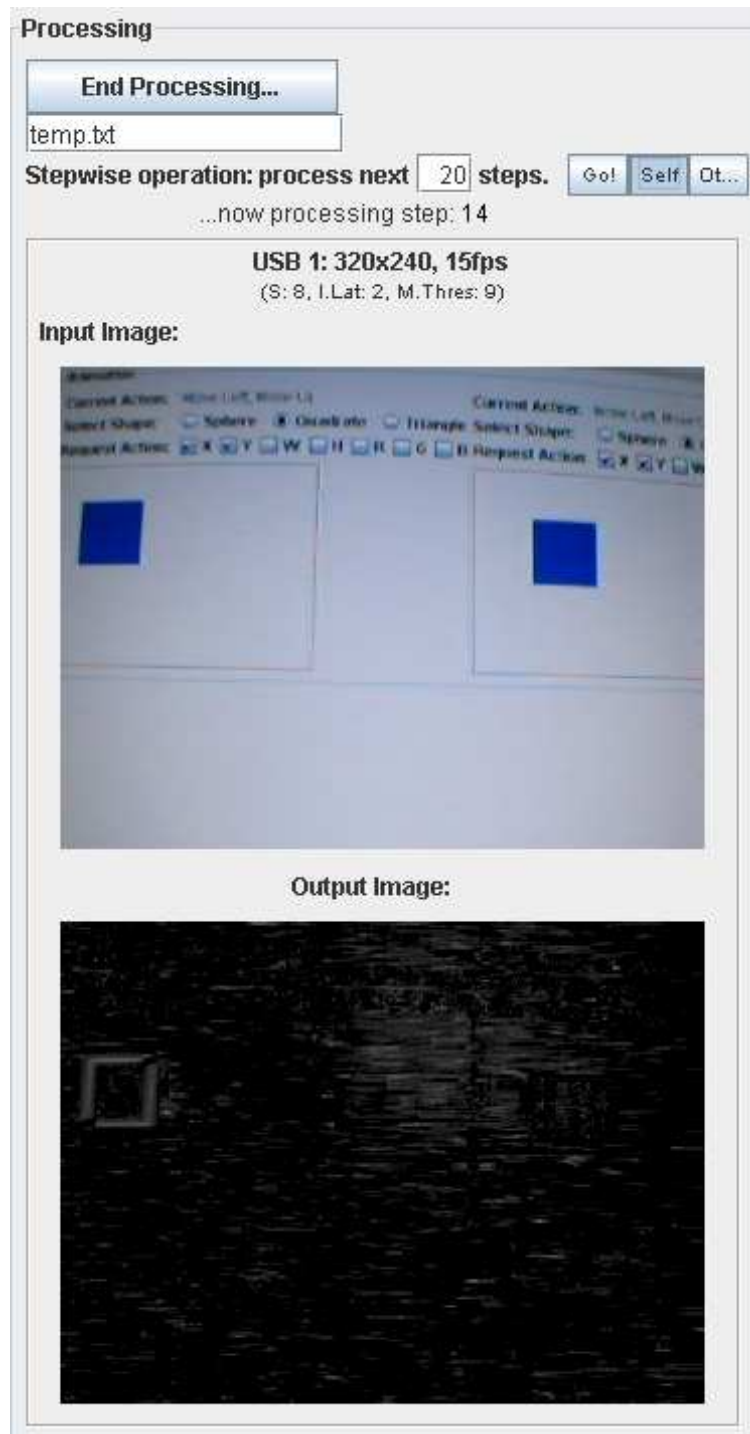


Figure 2.5: Self Moving, Other Stationary - Effect Of Noise

## Chapter 3

# Self-Other Discrimination Using Optic Flow

The algorithm we chose for comparison with HM was Fitzpatrick's method (Fitzpatrick & Metta, 2003). In this chapter we present the design for the implementation of the Fitzpatrick method which uses optic flow analysis and the results of evaluating the same. Our implementation is not an exact implementation of the Fitzpatrick method, we have adapted it to suit our simulation while keeping the basic idea of performing optic flow analysis intact. We do not use proprioceptive information like joint angles, instead, we use efference copy (i.e., a copy of the commands). Also, instead of predicting arm location, we classify all the observed changes in the visual image with respect to the relationship with the agent (i.e., the self entity). Henceforth we will refer to our implementation as the modified Fitzpatrick implementation.

For the implementation of our self-other standard task based on the Fitzpatrick method, we required an agent capable of moving objects with a virtual arm, and observing the visual changes. The “self” referred to is the agent itself, and the “other” is anything in the world which is not related to or caused by the agent. The agent also needed a method of observing visual changes in the world, e.g., a camera. The simulated agent is an application which has two main parts:

- Physical simulation: Generates moving images on the computer screen, in accordance with commands sent to it.
- Optical module: Captures the images moving on the screen with a webcam pointed at the screen.

The agent is also capable of analysing the images from the webcam in order to find the correlation between observed motion from the optical module and actual motion of the arm. Figure 3.1 shows the experimental setup.

### **3.1 Physical World Simulation**

This component simulates the “self” and the “other” entities in the world. These entities are simulated as square shaped objects of the same size on the screen, capable of motion when commands are sent to them. Commands to move the two objects are accepted via the keyboard or automated using a starting position from a text file. Each object is

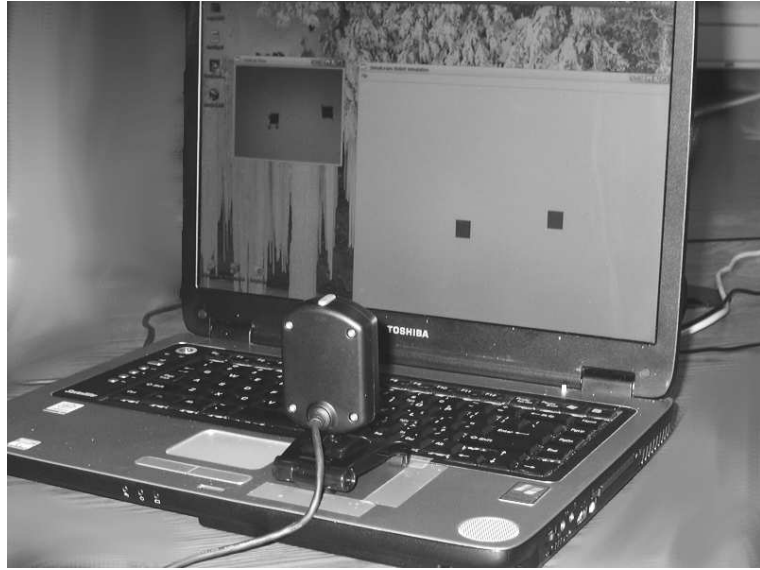


Figure 3.1: Experimental Setup - Modified Fitzpatrick method

capable of moving in either the horizontal or vertical direction only (not diagonally) per unit time. In the keyboard input mode, the self and the other object are randomly drawn on the screen, i.e., their coordinates in the application window are randomly generated, constrained by the size of the drawing area. In the automated mode, using the starting position from the text file, the self moves to the nearest edges in the horizontal and vertical direction using a predefined step size and in a predefined manner. The step size, the sizes of the objects and the shapes of the objects were decided after preliminary experimentation. It was observed that larger objects and larger step sizes resulted in less noise in optical flow. The square shape was chosen since optic flow vectors can be easily identified at the corners of the square by the optic flow algorithm we are using (see next section). The optic flow algorithm as used here does have dependencies on visual object

shape. The method generally depends on specific visual features available for tracking, and the specific feature detection algorithm used here (Shi & Tomasi, 1994) depends on corners. Within a drawing area of 2048x1024 pixels, the object size was 250x250 pixels and the step size was 50 pixels. This drawing area was scaled down by the QT canvas widget to 394x172 pixels, with the object size reduced to 40x44 pixels and a step size of 7. Any further references to dimensions in this chapter will use the programmed scale and not the actual size, since the actual size is dynamic and depends on the size of the application window. The motion of both objects is constrained by the dimensions of the drawing area they are in. The data collected from the physical simulation is a “self vector” composed of:

1. *magnitudeSelf*: Magnitude of motion (pixels) of the self.
2. *directionSelf*: Direction of motion (radians) of the self.

Our operating system for this experiment was Microsoft Windows XP. For the physical world simulation, we used QT, a C++ cross-platform application development framework<sup>1</sup>.

## 3.2 Optical Sensor

This component functions as the visual sensor of the agent. The webcam used for the experiment was a Creative NX Ultra (Creative Labs, California) working under Microsoft Windows XP. To assure synchronization between the motion of the “self” object and the

---

<sup>1</sup><http://www.trolltech.com>

images captured by the optical sensor, the optical sensor acquires an image from the camera each time the “self” object or the “other” object moves. As described earlier, this is necessary because the windowing system in an operating system is not synchronous. We followed a technique of drawing the object and then waiting some time for the drawing to complete before we acquired an image from the camera. The resolution of the captured images was 320x240 pixels. Images were captured from the camera and manipulated using Intel OpenCV<sup>2</sup>. OpenCV reads in the image from the camera and provides a left-to-right inverted image to the programmer. The algorithm used to compute optic flow vectors is Lucas and Kanade (1981). We collect optic flow vectors consisting of:

1. *magnitudeObserved*: Magnitude of motion (real number) as observed on the screen.
2. *directionObserved*: Direction of motion (radians) as observed on the screen.

The number of features tracked by the algorithm are set to 8. The corners of the self and other objects are tracked by the algorithm. Each tracked feature results in a separate optic flow vector. The camera observed the drawing area only and not the entire application window. This was necessary since the algorithm frequently identified application window edges as interesting features to track, ignoring some object corners.

---

<sup>2</sup><http://sourceforge.net/projects/opencvlibrary/>

### 3.3 The Need for Learning

As described above, the agent has the capacity to move its virtual arm on the screen, and also has the capacity to observe this motion using the webcam. This observed motion is then processed to obtain optic flow vectors which give the direction of the motion on the screen. After obtaining the optic flow vectors the agent needs to determine if this observed motion is related to the self vector. We are using the self vector and not the commands sent to move the self because the self vector is analogous to efference-copy and provides the same information in a format similar to the optic flow vectors (i.e., magnitude and direction instead of left, right, up or down).

But how can we find the connection between the optic flow vectors and the self vector? That is, how do we discriminate between optic flow vectors caused by self vs. those caused by the other? In general, a simple mathematical function (e.g.,  $OpticFlowVector = \alpha * SelfVector$  where  $\alpha$  is a constant) cannot suffice for this purpose. For example, if the virtual arm is pushing against a boundary of the drawing area, the optic flow vectors would not change, since the virtual arm cannot move beyond the bounds of the drawing area. Also, the optic flow vectors observed are not always perfect, i.e., a movement in the horizontal direction to the right does not always generate an optic flow vector which points exactly in the opposite direction. Our solution is to algorithmically learn the relation between the optic flow vectors and the self vectors. To this end, we have trained a neural network to perform this association. When the optic flow vector is caused

by the movement of the virtual self, the two vectors are said to be *related*.

### 3.3.1 Structure of neural network

Our goal is to classify each of the optic flow vectors as either arising from self or other. The neural network we used for this purpose is a fully connected feed forward neural network, trained with an implementation of backpropagation (Rumelhart, Hinton & Williams, 1986) developed in our laboratory. The network has 4 inputs and 1 output, with one hidden layer. The structure of the network is shown in Figure 3.2.

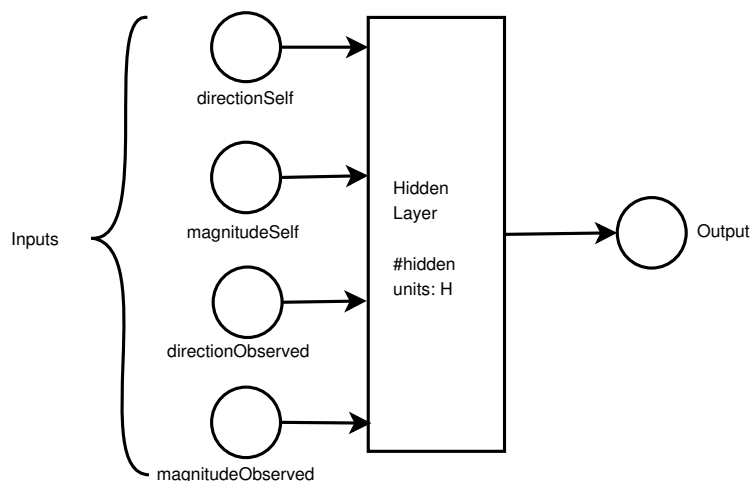


Figure 3.2: Structure Of Neural Network

The inputs are:  $\{directionSelf, magnitudeSelf, directionObserved, magnitudeObserved\}$ , with the expected output being either 1 or 0. An output of 1 indicates that the optic flow vector resulted from the self motion, i.e. the self vector and the optic flow vector *are*

related. An output of 0 indicates that the self vector and the optic flow vector are *not* related. This means that by chance, the network can correctly discriminate self from the other with 50% accuracy. Sample input data to the network is shown in Table 3.1. The first example in Table 3.1 shows a self vector and an optic flow vector that are related. In this example, the directions of the self and the optic flow vectors are relatively close, as are the magnitudes. This indicates that the two are related. Note that the self and the optic flow vectors are in the same direction instead of opposing directions, this is because OpenCV flips the image left-to-right while reading it from the camera. The second example in Table 3.1 shows an example where the optic flow vector and the self vector are not related. In this example, neither the directions nor the magnitudes of the two vectors are particularly close to each other.

<i>Example</i>	<i>Direction Self</i>	<i>Magnitude Self</i>	<i>Direction Observed</i>	<i>Magnitude Observed</i>	<i>Expected Output</i>
1	3.14159	5	2.97644	6.08276	1
2	3.14159	5	0.785398	0.41421	0

Table 3.1: Sample Training Data

### 3.3.2 Generation of training data

All data collected was under uniform lighting, to avoid changes in the visual images due to changes in illumination. We also taped the camera to fix its position prior to the start of any data collection, to ensure that the setup did not change from one data collection session to another. To train the network, data was collected by running the simulation in automated mode. The starting locations for the “self” and the “other” objects were read from a text file. For each of the self and other positions, data was collected from 9 starting positions, as illustrated in Figure 3.3. Figure 3.3 indicates one half (left or right) of the drawing area. The camera viewed both the self object as well as the other object, and 8

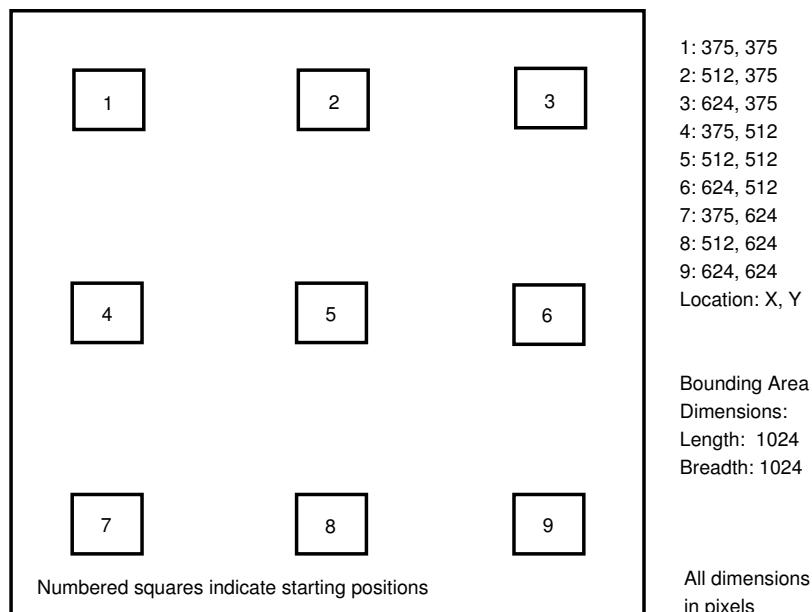


Figure 3.3: Starting Positions For Automated Data Collection

features were being tracked in the optic flow computation. This resulted in 8 optic flow

vectors and 1 self vector collected per unit time. Since the Lucas and Kanade (1981) optic flow algorithm first identifies significant features before looking for optic flow, the eight optic flow vectors were evenly distributed amongst the “self” and the “other” objects. The corners of each square object were identified as the significant features to track in the image by the optic flow algorithm. Figure 3.4 shows the optic flow vectors generated when the “self” object is moving. We did *not* collect training data for Case 3, where the self is moving and the other moves simultaneously in a random direction. This case was left for evaluation purposes after the neural network was trained. In Figure 3.4, the optic flow

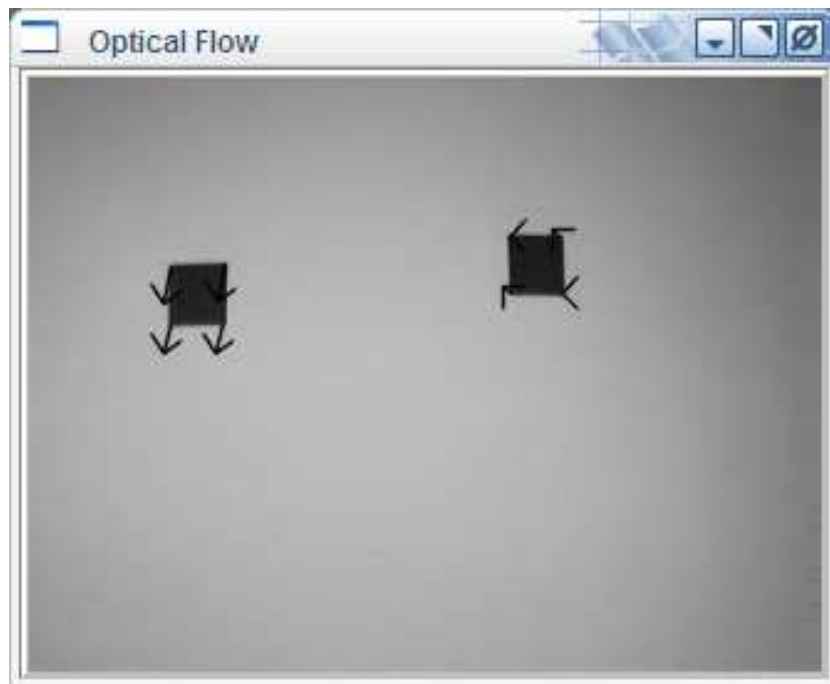


Figure 3.4: Sample Optic Flow Vectors, Self Moving Downwards

vectors at the corners of the “self” object should be classified as related and the optic flow

vectors at the corners of the “other” object should be classified as not related.

To facilitate automatic classification of the training data since we are using a supervised training technique (Rumelhart, Hinton & Williams, 1986), there was an invisible boundary beyond which the “self” object could move right and the “other” object could not move left. This is illustrated in Figure 3.5 which is a screenshot of the application drawing area with the invisible boundary superimposed on the image. Thus, of

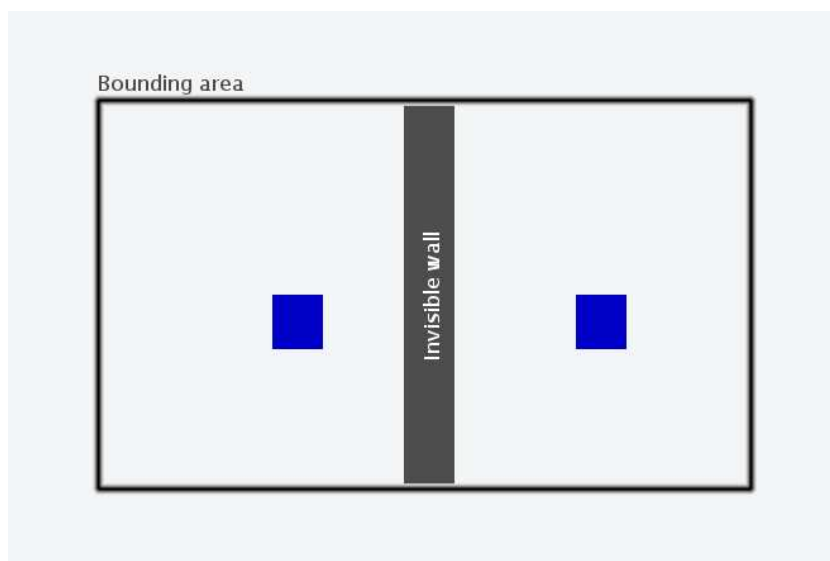


Figure 3.5: Invisible Boundary Between Self And Other

the 8 optic flow vectors being generated per unit time, those on the left half of the image were classified, for training purposes, as related (expected output = 1) while those on the “right” were classified, also for training purposes, as not related (expected output = 0).

Data gathered from 7 of the 9 starting locations was used for training the network, one was kept aside for validation and one for testing. The number of samples in the training set was

<i>Dataset</i>	<i>Number</i>	<i>Related</i>	<i>NotRelated</i>
Training	6221	2876	3345
Validation	996	441	555
Testing	749	320	429

Table 3.2: Distribution Of Training, Validation And Testing Datasets For Cases 1 & 2

6221, the number of samples in the validation set was 996 and the number of samples in the testing set was 749. Table 3.2 gives the related and non-related breakdown. The number of samples in a dataset is not perfectly divisible by 8 because only part of the data from Case 2 was used for training<sup>3</sup>. To compensate for the amount of time taken by the graphics windowing system of the underlying operating system, there was a delay of 2 seconds between each move. This was necessary because rapid redrawing of images froze the application, and also because drawing of images on the screen by the windowing system is not synchronous, i.e., there may be delays in drawing the shape on the screen after the drawing command is issued.

---

<sup>3</sup>For Case 1 (self moving, other stationary), of the 8 vectors collected per unit time, 4 were related and 4 unrelated. In Case 2 (self stationary, other moving), all vectors collected were unrelated. Training the network on the entire dataset collected skewed the output of the network towards unrelated, since more unrelated training samples were present in the entire dataset. Hence we only used part of the data from Case 2.

### 3.3.3 Training and testing the network

Now that the data had been generated, we tried to fit a network to the training data. To avoid overfitting, we used the validation set and the following stopping criteria for training. When the performance of the network deteriorated (i.e., the error over the validation set increased over a window size of 2 epochs) or when the error difference between two successive epochs was less than 0.001, we stopped training. We used one hidden layer with 1, 2, 3, 5 or 10 hidden units and experimented with the learning rate (range: 0.001 - 0.2) and the training mode (batch or incremental training) parameters. Figure 3.6 shows the range of the error values over the different parameter sets, sorted by error. The results of

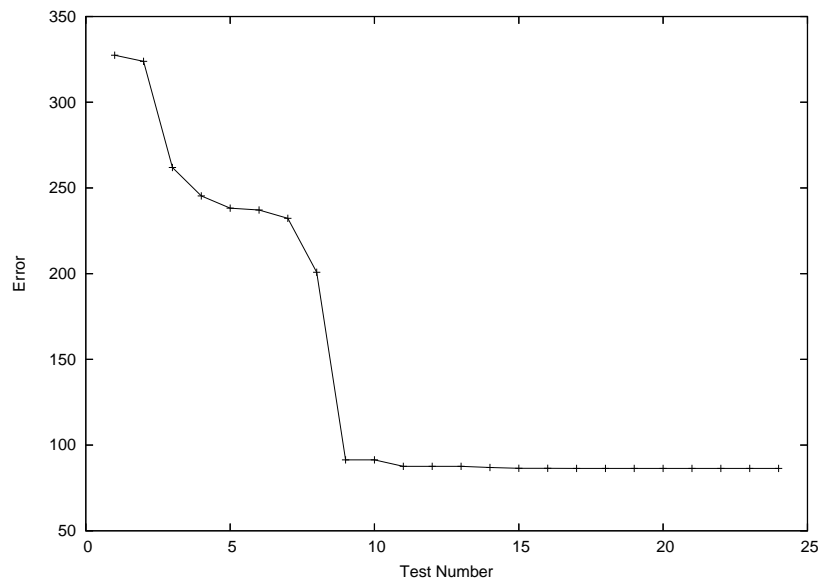


Figure 3.6: Error Range Over Different Parameters

training the network as evaluated on the testing dataset (last row, Table 3.2) are detailed in Table A.1 (see Appendix). For convenience, we present the first (best performance) row

of Table A.1 in Table 3.3. The *Error* column in Table 3.3 is the distance from the actual

<i>H</i>	<i>LR</i>	<i>Mode</i>	<i>Error</i>	<i>#Correct</i>	<i>%Correct</i>	<i>P – value</i>
2	0.2	0	86.3619	659	87.8666	<0.0000001

Table 3.3: Results Over Best Performing Parameter Set

network output values and is computed using Equation 3.1. The column labeled *P-value* gives the p-value computed by applying the cumulative binomial to the discrimination results.

$$Error = \sum_{allOpticFlowVectors} |ExpectedValue - ComputedValue| \quad (3.1)$$

All results have been averaged over six repetitions of running the network with the same parameters.

As seen in Table A.1, the best training results were observed with 2 hidden units, a learning rate of 0.2 and training in incremental mode. For this network, the percentage of related vectors identified correctly is 96.26% and the percentage of unrelated vectors identified correctly is 81.59%. The accuracy results are listed in Table 3.4. Further analysis

	Overall	Related	Unrelated
Accuracy (%)	87.87	96.26	81.59

Table 3.4: Breakdown Of Best Results For Cases 1 & 2

of the results showed that within the unrelated set, the accuracy for Case 2 (when self

vector = 0, 0) is 33%. This can be improved by training with more samples for this particular case, however, the performance of the network then deteriorates Cases 1 and 3. The structure of the best performing neural network is shown in Figure 3.7.

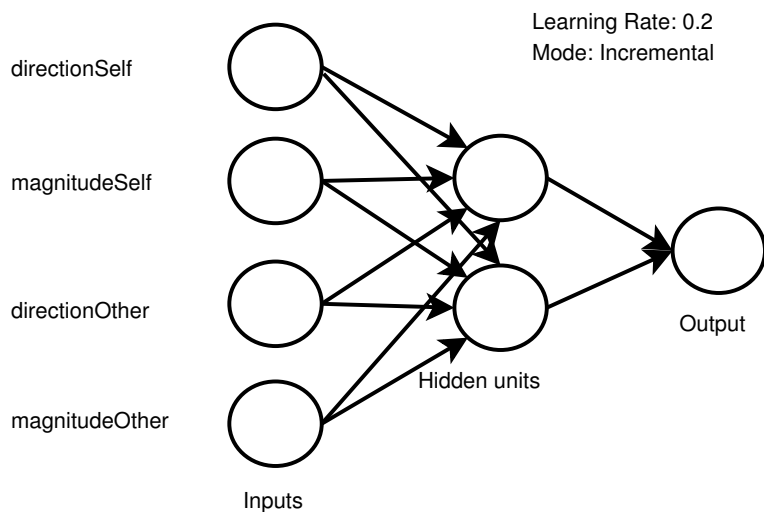


Figure 3.7: Final Network Architecture

To understand the statistical significance of the results observed, we computed p-values for the results of each network. The null hypothesis states that the results are due to chance. The column *P-value* in Table 3.3 shows the p-value for the best performing network and this value is less than 0.05. Thus we can reject the null hypothesis with 95% confidence and state that the results observed are not due to chance. As seen in Table A.1 the null hypothesis can be rejected even for the worst performing network. Thus it appears clear that the networks have learned this self-other discrimination.

The testing data collected above accounts for Case 1 and Case 2 (self moving and other

stationary, and self stationary and other moving). Now we needed to evaluate Case 3 (both self and the other move). We collected 5616 samples (702 steps) by running the simulation under control of the keyboard, with the movement of the self controlled by the keyboard and the other moving according to a random process. In this case the network was able to classify the optic flow vectors as arising from self or other with 74.34% overall accuracy. The breakdown of the result is listed in Table 3.5. Since the objects could move in only four directions, the probability of the direction of the other movement being the same as the direction of the self is 0.25. Since our previous accuracy in classifying unrelated vectors with the best network was 81.59%, given a 25% chance of getting an unrelated vector which is similar to the related vector, the best result we would now expect would be an accuracy of 61.12%. We applied the 2x2 Chi-square test to the *unrelated* results, to find

<i>Expected</i>	<i>Correct</i>	<i>Incorrect</i>	<i>Total</i>
Related	2628	180	93.59
Not related	1547	1261	55.09

Table 3.5: Results Distribution, Self And Other Moving

out if the unrelated results in Case 3 were due to chance, given the unrelated results in the previous cases. Table 3.6 shows the values used in the Chi-square computation. Since the p-value computed ( $p < 0.001$ ) is less than 0.05, we can reject the null hypothesis that the two results distributions are not related and conclude that it is unlikely that the self-other discrimination results we observed were due to chance.

	Unrelated, Case 3	Unrelated, Case 1 & Case 2
Correct	1547	351
Total	2808	430

Table 3.6: 2x2 Chi-square, Unrelated Vectors

### 3.3.4 Discussion

The goal of this chapter was to describe and evaluate our implementation of the modified Fitzpatrick method. In the self-other discrimination task, both the self and other objects were modeled as shapes on a computer screen, and a webcam served as the visual sensor for the system. Optic flow vectors computed using the Lucas and Kanade (1981) algorithm were classified as either arising from the “self” or the “other” object using a feedforward neural network trained with backpropagation (Rumelhart, Hinton & Williams, 1986). The inputs to the neural network were the optic flow vector and the self vector, and the output, a value between 0 and 1, classified the two vectors as either related (arising from self) or not related (arising from other). The best performing network had 2 hidden units, a learning rate of 0.2 and was trained in incremental mode. For this network, the overall average accuracy in discriminating self from the other was 76.85% (averaged across Tables 3.3 and 3.4).

Analysis of the results showed that the network faired poorly when the self vector was 0 (i.e., the self object was not moving). An attempt to improve the performance of the

network for this case worsened the performance of the network in classifying the other cases. A question to ask at this point is: Is there a need for a neural network in the case where the self is not moving? The rationale for using a neural network was that a simple mathematical function could not be used to classify the optic flow vectors. However, in this case (Case 2), when the self is not moving any optic flow vectors observed are definitely not caused by self and hence can be directly classified as unrelated. The steps followed to classify optic flow vectors in such a scenario are described in Figure 3.8.

```
//begin classification  
  
if (self vector = {0, 0})  
  
    then output = 0 //classify as not related  
  
else  
  
    output = network_classify(self vector, optic flow vector).  
  
    if (output  $\geq$  0.5) output = 1 //classify as related  
  
    else output = 0 //classify as not related  
  
//end classification
```

Figure 3.8: Modified Classification Algorithm

# Chapter 4

## Discussion

In this thesis we set out to compare two synchrony detection algorithms, using a standard task. The criteria chosen for comparison was self-other discrimination, i.e., we were testing the ability of each method to distinguish between motion caused by the “self” and motion caused by the “other”. This ability to do self-other discrimination was tested in the following three cases:

- Self moving, other stationary.
- Self stationary, other moving.
- Self moving, other moving.

Of the two methods we chose for comparison, the first was an extension of the Helder (2003) implementation of the Hershey and Movellan (2000) algorithm, and the second was

a method based on optic flow analysis developed by Fitzpatrick (2003). The extended Helder (2003) implementation computed synchrony between two streams of information: visual images and commands sent to move objects on the screen. The extent of synchrony was used to discriminate between self and other, based on a summation technique developed in our laboratory (Prince, *et al.*, 2004). The modified Fitzpatrick (2003) implementation used optic flow analysis and a trained neural network to discriminate between self and other. Each optic flow vector was classified as either related or unrelated.

## 4.1 Comparison

Table 4.1 lists the results for each method for all three cases. The HM results are for the best performing technique which used Gaussian filtering and Sobel edge detection, and the results for the modified Fitzpatrick implementation are for the best performing neural network. The percentages shown in the table are the overall average accuracy of each method for that particular case.

### 4.1.1 Self moving, other stationary

In the case of the self moving and the other stationary, both methods performed reasonably well and were able to classify the output as related to self with a higher percentage than classifying randomly with 50% accuracy. The optic flow method performed better than the method based on HM. However, it should be noted that in the

Case	Hershey & Movellan Implementation	Modified Fitzpatrick Implementation
Self moving, other stationary	71.67	87.87
Self stationary, other moving	100	33
Self moving, other moving	56.67	74.34

Each value indicates the accuracy in percentage.

Table 4.1: Summary Of Results

optic flow method, optic flow vectors are computed at the corners of the “other” object, even though it is not in motion. That is, no zero optic flow vector was ever computed. Thus we get optic flow vectors which are attached to the other object, which are classified as not related. HM does not do such a classification, while synchrony computation is performed for all pixels in the image, only the self object is outlined. Thus the other object is not identified and later labeled as unrelated. However, accuracy of the HM method was not as high due to noise in the mixelgram output.

#### 4.1.2 Self stationary, other moving

HM handles the self stationary, other moving case very easily, since with constant self, all mixels will be zero. Hence all movement on the screen is classified as not related. The neural network in our modified Fitzpatrick method however does not perform well in this

situation. This can be rectified by applying the modified classification algorithm sketched out in the discussion section of Chapter 2. In fact, the modified classification algorithm will then behave like HM is behaving in this situation. If the modified classification technique in Figure 3.8 is used, then the modified Fitzpatrick method would perform better than the HM method due to a lack of a window. HM would lag behind if the self started moving, such a change would immediately be detected by the optic flow method which operates on successive frames.

### **4.1.3 Self moving, other moving**

For this case, the modified Fitzpatrick implementation outperforms HM. The results from testing the HM implementation for this case do not demonstrate a high level of self-other discrimination, when compared with the modified Fitzpatrick implementation.

### **4.1.4 Self stationary, other stationary**

Apart from the 3 cases we have considered, a fourth case exists where both self and other are stationary. While we did not explicitly test this case, using both the modified Helder (2003) implementation and Figure 3.8 in the modified Fitzpatrick implementation we expect motion observed (if any) in the scene to be classified as not related to the self. In the modified Helder (2003) implementation, the Case 4 frames would be classified as not related, and in the modified Fitzpatrick implementation, the optic flow vectors observed

would be classified as not related using the technique from Figure 3.8.

## 4.2 Conclusion

Our experiments with both methods show that both performed well in detecting a “self” relationship, but the Fitzpatrick method was able to better discriminate between the self and the other under our test conditions. However, the requirement that the neural network needs to be pretrained is a disadvantage which HM does not suffer from. This could be overcome if training the network could be done in a dynamic on-the-fly manner. In the remainder of this chapter we will discuss some of the limitations of the Fitzpatrick method and some techniques to work around these limitations.

## 4.3 Limitations of the Fitzpatrick method

The Fitzpatrick method has certain limitations, some of these are due to our implementation and some inherent to the algorithm.

- Our neural network method is currently unable to handle wall boundaries. Thus, if the self object were to hit a wall and stop moving or bounce back, the neural network would be unable to classify the optic flow vectors. This may be remedied by using a more complete set of input values to the neural network, such as the distance from the object to the wall.

- Since the network is pretrained for a certain setup, what happens if the self-related vector relationship changes? This relationship can change due to some change in the experimental setup, such as camera angle or camera magnification. In our current implementation, the neural network would need to be retrained. This argues for ongoing adaptation, and using a more fully featured proprioceptive information set or effector-copy information. This information could be features like the distance of the camera to the screen, camera angles, or (if implemented on a robot) the joint angles for the robot's arm. In our current technique, we use the artificial construct of an invisible wall to collect and label training data, thus employing a supervised learning approach. However, any changes in the setup (such as a change in camera angle) may cause this construct to fail, since the invisible wall may no longer be at the center. We cannot train the network for all possible setups, we need to train the network as and when the scenario changes. And since an artificial construct to label training data may not always be feasible, this implies that we need to use an *unsupervised* learning technique.
- The Fitzpatrick method is also susceptible to noise in the form of random optic flow vectors computed for non-moving objects. No zero optic flow vectors were computed by the optic flow algorithm, even when no motion was visible in the scene. In such a situation, random optic flow vectors were computed. This could be resolved by processing the input data streams prior to classification using a technique like frame subtraction. Frame subtraction is a technique where the pixel values in one image are

subtracted from the pixel values in another image, giving an image which shows the difference between the two images.

## 4.4 Future Work

Based on our experience with the two methods, we have a few suggestions which may potentially remove the limitations and improve performance of the two methods.

- For both the HM and the Fitzpatrick methods, frame subtraction before analysis may help reduce noise. Applying frame subtraction to the Fitzpatrick method would remove the “other” object, for example, from the analysis if it is not moving. Thus, all points chosen for tracking by the optic flow method would be associated with the self object only. This would remove the random optic flow vectors that are currently associated with the “other” object even when it is not moving.

Frame subtraction could also help to reduce noise in the Hershey and Movellan method by outlining only the moving part of the image. The remaining pixels would then get a zero synchrony value assigned to them.

- Exploring other optic flow methods may improve the performance of the Fitzpatrick method. The Lucas and Kanade (1981) algorithm is a feature based method which computes the optic flow at a small number of well-defined image features. In our implementation, these features are the corners of the objects. Other possible methods are gradient based optic flow computation (e.g., Horn & Schunck, 1981 and Nagel,

1990) and correlation based optic flow computation (e.g., Burt, Bergen, Hingorani, Kolczynski, Lee, Leung, Lubin & Shvaytser, 1989). However, one point to note is that using gradient based methods would be computationally very intensive, since these methods estimate optic flow at every point in the image. Correlation based methods compare parts of the image in terms of brightness patterns and may provide a good alternative.

- As mentioned in Section 4.2, there are some limitations to the neural network. To remove this requirement, we propose two alternative ways of training the neural network:

### **Method 1**

This method proposes that we collect training data for the related and unrelated vectors separately, in a slightly different, artificially enhanced manner. This should do away with the awkward construct of an invisible wall as well as reduce noisy data. The steps to collect training data are:

1. *Collecting data for the related vectors*

Move the self object only, and then use frame subtraction prior to computing optic flow vectors. The frame subtraction will ensure that the only significant features to track will be related to the self object only, and restricting the number of features to track will increase the chance of only the self being tracked. Collecting optic flow vectors in this manner would help the neural network to learn the related vectors.

## 2. *Collecting data for the unrelated vectors*

Move the self object internally, without moving it on the screen. The other also moves randomly, but not with the same magnitude and direction as the internal movement of the self. Such a situation has a biological metaphor, e.g., children would have some variability to their internal state which does not necessarily translate to an externally visible change. This will ensure that the self and the other movements are do not coincide, and avoid collection of noisy training data. By not moving the self on the screen, we will ensure the other is the only moving object in the image. Using frame subtraction will ensure that the other is the only significant object in the scene, similar to the above point. The optic flow vectors collected in this situation would unrelated, and help learn that relationship. The problem with this is that the movement of the other should be comprehensive enough to learn the relationship.

After training, the network should be able to classify any optic flow vectors as either related or not related based on the relationship learned. The frame subtraction should remove the “no zero optic flow vector” effect, by removing those features from the features to be tracked. It should also remove the necessity of the “invisible wall” during training data collection since only related or unrelated optic flow vectors will be collected per unit time.

### **Method 2**

This method suggests that a neural network be used to predict optic flow vectors

arising from movement of the self instead of classifying observed optic flow vectors as related or not related. The neural network takes as input a self vector and outputs a prediction for the expected optic flow vectors. The procedure suggested is as follows:

1. Move the self object.
2. Use the self vector to predict the expected optic flow vectors, i.e., the self vector is the input to the network.
3. Compute error between the prediction and the actual optic flow vectors.
4. Disregard the optic flow vectors with the worst errors, i.e., the bottom  $N\%$ . Our hypothesis for using the vectors with the least errors as related is that movements related to self should be more predictable. That is, based on the self vector, we should be able to predict self motion more easily than other motion.
5. Train with the vectors which have the least errors, i.e. use these vectors as expected output.

This process may help the the neural network to learn the vectors with relatively small errors quickly. The data which generated the maximum errors will be ignored. Since this process does not classify the output and instead predicts the optic flow vectors, the predicted optic flow vectors can be used to compute where the self object has moved. Hypothetically, in this method, the training might be never stopped. In fact, the percentage of vectors discarded might be reduced. This may broaden the scope of the network and help learn relationships which have an indirectly contingent

relationship. One possible learning algorithm that can be used here is the IHDR technique of Weng and Hwang (2003). This method operates incrementally, and so would be suited for ongoing adaptation.

In closing, we mention that the developmental approaches to robotic self-other discrimination have recently started becoming a topic of interest to other researchers. Other than our own, one of the first research efforts in developmental robotic self-other discrimination, that we are aware of, was conducted by Michel, Gold and Scassellati (2004; see also Gold & Scassellati, 2005). This work used the latency between a robots actions and the result observed in the visual sensors to acquire self-other discrimination skills. Furthermore, in two very recent conferences (held in July 2005), the number of entries into this new topic has blossomed (Edsinger-Gonzales, 2005; Lungarella & Sporns, 2005; Olsson, Nehaniv, & Polani, 2005; Yoshikawa, Yoshimura, Hosoda, & Asada, 2005).

# Appendix A

## Tables

<i>H</i>	<i>LR</i>	<i>Mode</i>	<i>Error</i>	<i>#Correct</i>	<i>%Correct</i>	<i>P – value</i>
2	0.2	0	86.3619	659	87.8666	<0.0000001
2	0.125	0	86.3711	659	87.8666	<0.0000001
1	0.125	0	86.3716	659	87.8666	<0.0000001
2	0.1	0	86.4067	659	87.8666	<0.0000001
5	0.1	0	86.4067	659	87.8666	<0.0000001
1	0.1	0	86.4068	659	87.8666	<0.0000001
10	0.1	0	86.4071	659	87.8666	<0.0000001
3	0.1	1	86.4071	659	87.8666	<0.0000001
1	0.075	0	86.4777	659	87.8666	<0.0000001
2	0.075	0	86.4782	659	87.8666	<0.0000001
1	0.025	0	86.95	659	87.8666	<0.0000001
3	0.01	0	87.5759	659	87.8666	<0.0000001
2	0.01	0	87.5761	659	87.8666	<0.0000001
1	0.01	0	87.5766	659	87.8666	<0.0000001
2	0.001	0	91.3571	659	87.8666	<0.0000001
1	0.001	0	91.3574	659	87.8666	<0.0000001
3	0.1	1	200.8196	558	74.4	<0.0000001
2	0.01	1	232.2833	565.3333	75.3777	<0.0000001
1	0.1	1	237.1433	519.3333	69.2444	<0.0000001
3	0.01	1	238.2013	537.5	71.6666	<0.0000001
2	0.1	1	245.3611	501	66.8	<0.0000001
1	0.01	1	261.9448	519	69.2	<0.0000001
2	0.001	1	323.851	482.8333	64.3777	<0.0000001
1	0.001	1	327.4116	409.5	54.6	0.007186

*#Inputs: 4 #Outputs: 1 #Testing: 750*  
*H: #Hidden units LR: Learning Rate*  
*Mode: 0 – Incremental, 1 – Batch*

Table A.1: Training Over Different Parameters

# Bibliography

Arsenio, A. & Fitzpatrick, P. (2003, December 15–18) *Exploiting cross-modal rhythm for robot perception of objects*. Presented at the 2nd International Conference on Computational Intelligence, Robotics, and Autonomous Systems, Singapore.

Boykov, Y. & Kolmogorov, V. (2001). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition, volume 2134 of LNCS*, 359–374.

Brooks, R. (1999). *Cambrian Intelligence: The Early History of the New AI*. Cambridge, MA: MIT Press.

Burt, P.J., Bergen, J.R., Hingorani, R., Kolczynski, R.J., Lee, W.A., Leung, A., Lubin, J., Shvayster, J. (1989). Object tracking with a moving camera; an application of dynamic motion analysis. *In IEEE Workshop on Visual Motion*, held at Irvine, CA.

- Covell, M & Bregler, C. (1996). Eigenpoints. *Proc. Int. Conf. Image Processing, Vol. 3*, 471–474.
- Edsinger-Gonzales, A. (2005). Developmentally guided ego-exo force discrimination for a humanoid robot. *Proceedings of the Fifth International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*.
- Fitzpatrick, P. (2003). *From First Contact to Close Encounters: A Developmentally Deep Perceptual System for a Humanoid Robot*. Unpublished doctoral dissertation, Massachusetts Institute of Technology, Artificial Intelligence Laboratory. MIT AI Technical Report 2003–008, June 2003.
- Fitzpatrick, P. & Metta, G. (2003). Grounding vision through experimental manipulation. *Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences*, 36, 2165–2185.
- Gergely, G. & Watson, J. S. (1999). Early socio-emotional development: Contingency perception and the social-biofeedback model. In: P. Rochat (Ed.), *Early Social Cognition: Understanding Others in the First Months of Life* (pp. 101–136) Mahwah, NJ: Lawrence Erlbaum Assoc.
- Gogate, L. J. & Bahrick, L. E. (1998). Intersensory redundancy facilitates learning of arbitrary relations between vowel sounds and objects in seven-month-old infants. *Journal of Experimental Child Psychology*, 69, 133–149.

Gold, K. & Scassellati, B. (2005). Learning about the self and others through contingency. *Developmental Robotics AAAI Spring Symposium*.

Helder, N. A. (2003). A real-time, computational model of perceptually-based contingent behavior detection. *Honors Project, University of Minnesota Duluth, Department of Computer Science*.

Internet: <http://www.cprince.com/projects/KidCause/Detect/>

Hershey, J. & Movellan, J. (2000). Audio-vision: Using audio-visual synchrony to locate sounds. In S. A. Solla, T. K. Leen, & K. -R. Müller (Eds.). *Advances in Neural Information Processing Systems 12* (pp. 813–819). Cambridge, MA: MIT Press.

Horn, B.K.P & Schunck, B.G. (1981). Determining optical flow. *Artificial Intelligence, 17*, 185–203.

Kuffner J. J., Nishiwaki K., Kagami S., Kuniyoshi Y., Inaba M., & Inoue H. (2002). Self-collision detection and prevention for humanoid robots. *In Proceedings at the IEEE International Conference on Robotics and Automation (ICRA '2002)*, 2265–2270.

Lucas, B. and Kanade T. (1981). An iterative image registration technique with an application to stereo vision. *Proc. DARPA Image Understanding Workshop*.

Lungarella, M., Metta, G., Pfeifer, R. & Sandini, G. (2003). Developmental robotics: A survey. *Connection Science, 15*, 151–190.

Lungarella, M. & Sporns, O. (2005). Information self-structuring: Key principle for learning and development. *Proceedings of the 4th IEEE International Conference on Development and Learning*, held at Osaka, Japan, July 19-21.

McGraw, M. B. (1945). *The Neuromuscular Maturation of the Human Infant*. New York: Columbia University Press.

Metta, G. & Fitzpatrick, P. (2003). Better vision through manipulation. *Adaptive Behavior*, 11, 109–128.

Metta, G., Sandini, G. & Konczak, J. (1999) A developmental approach to visually-guided reaching in artificial systems. *Neural Networks*, 12, 1413–1427.

Michel, P., Gold, K., & Scassellati, B. (2004). Motion-based robotic self-recognition. Presented at the *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Mislivec, E. J. (2004). *Audio-visual synchrony for face location and segmentation*. Undergraduate Research Opportunity Project, University of Minnesota Duluth [On-line]. Internet: <http://www.cprince.com/PubRes/SenseStream>

Morency, L.-P. & Gupta, R. (2003) Robust real-time egomotion from stereo images. *ICIP 2003*, 3, 719–722.

Nagel, H.H. (1990). Extending the ‘oriented smoothness constraint’ into the temporal domain and the estimation of derivatives of optical flow. *In Proc. 1st European Conf. on Computer Vision*, 139–148.

Needham, A. (2000). Improvements in object exploration skills may facilitate the development of object segregation in early infancy. *Journal of Cognition and Development*, 1, 131–156.

Olsson, L., Nehaniv, C. L., & Polani, D. (2005). From unknown sensors and actuators to visually guided movement. *Proceedings of the 4th IEEE International Conference on Development and Learning*, held at Osaka, Japan, July 19-21.

Prince, C. G., Helder, N. A., & Hollich, G. J. (2005). Ongoing emergence: A core concept in epigenetic robotics. *Proceedings of The Fifth International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*, held at Nara, Japan, July 22-24, 2005.

Internet: <http://www.cprince.com/PubRes/EpiRob05/>

Prince, C. G., Helder, N. A., Mislivec, E. J., Ang, B. J., Lim, M. S., & Hollich, G. J. (2003). Taking contingency seriously in sensory-based models of learning in infants. *Poster presented at the 2003 Meeting of the Cognitive Developmental Society*, held at Park City, UT, USA, October 24-25.

- Prince, C. G., Hollich, G. J., Helder, N. A., Mislivec, E. J., Reddy, A., Salunke, S. & Memon, N. (2004). Taking synchrony seriously: A perceptual-level model of infant synchrony detection. *Presented at The Fourth Annual Workshop on Epigenetic Robotics*, held at Genoa, Italy, August 25-27.
- Rabiner, L. & Juang, B. (1993). *Fundamentals of Speech Recognition*. Prentice Hall.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533–536.
- Shi, J. & Tomasi, C. (1994) Good features to track. *IEEE Conference on Computer Vision and Pattern Recognition*, held at Seattle, U.S.A., 593–600.
- Slaney, M. & Covell, M. (2001). A linear operator for measuring synchronization of video facial images and audio tracks. *Proceedings of Neural Information Processing Society 13*, MIT Press.
- Sugamara, N. & Itakura, F. (1986). Speech analysis and synthesis methods developed at ECL in NTT - From LPC to LSP. *Speech Communications*, *5*, 199–215.
- Talukder, A. & Matthies, L. (2004). Real-time detection of moving objects from moving vehicles using dense stereo and optical flow. *Proceedings of the IEEE International Conference on Intelligent Robots and Systems* held at Sendai, Japan.

- Thelen, E. (2000). Grounded in the world: Developmental origins of the embodied mind. *Infancy, 1*, 3–28.
- Thelen, E., & Fisher, D. M. (1982). Newborn stepping: An explanation for a “disappearing reflex.” *Developmental Psychology, 18*, 760–775.
- Thelen, E & Smith, L. (1994). *A Dynamic Systems Approach to the Development of Cognition and Action*. Cambridge, MA: MIT Press. A Bradford Book.
- Vuppla, K. (2004). *Evaluation of Two Synchrony Detection Implementations*. Unpublished master’s thesis, University of Minnesota Duluth [On-line].  
Internet: <http://www.cprince.com/PubRes/VupplaThesis04>
- Webster’s Revised Unabridged Dictionary. (1996). [On-line].  
Internet: <http://www.dictionary.com>
- Weng, J. & Hwang, W. (2003). Online image classification using IHDR. *International Journal on Document Analysis and Recognition, 5*, 118-125.
- Weng, J., McClelland, J., Pentland, A., Sporns, O., Stockman, I., Sur, M., & Thelen, E. (2001). Autonomous mental development by robots and animals. *Science, 291*, 599–600.
- Yehia, H. C., Rubin, P. E., & Vatikiotis-Bateson, E. (1998). Quantitative association of vocal-tract and facial behavior. *Speech Communication, 26*, 23–44.

Yoshikawa, Y., Yoshimura, M., Hosoda, K., & Asada, M. (2005). Visio-tactile binding through double-touching by a robot with an anthropomorphic tactile sensor. *Proceedings of the 4th IEEE International Conference on Development and Learning*, held at Osaka, Japan, July 19-21.

Zlatev, J., & Balkenius, C. (2001). Introduction: Why 'epigenetic robotics'? *In Proceedings of the 1st International Workshop on Epigenetic Robotics*, held at Lund, Sweden, (pp. 1-4).