SENSORY INTEGRATION WITH ARTICULATED MOTION ON A HUMANOID ROBOT

JUAN L ROJAS

Thesis under the direction of Professor Alan R. Peters

The work in this thesis seeks to integrate the motion of a humanoid robot with its auditory and visual sensory information to achieve various reflex actions that mimic those of people. Such reflexes in the form of reach-grasp behaviors can enable the robot to learn through experience its own state and that of the world. A humanoid robot with auditory capabilities, stereo vision, and artificial pneumatic arms and hands was used to demonstrate tightly coupled sensory-motor behaviors in five different demonstrations. The complexity of succeeding demonstrations was increased to show that the reflexive sensory-motor behaviors combine to perform increasingly complex tasks. The humanoid robot executed these tasks effectively and established the ground-work for the further development of hardware and software systems, sensory-motor vector-space representations, and coupling with higher level cognition.

SENSORY INTEGRATION WITH ARTICULATED MOTION ON A HUMANOID ROBOT

By

Juan Luis Rojas

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

May, 2004

Nashville, Tennessee

Approved:

Professor Alan R. Peters II

Professor Mitch Wilkes

Para mi padre y madre que sacrificaron tanto por ver mis sueños cumplidos.

Para mis hermanos por su apoyo y amor.

Por mi Dios que todo le debo.

ACKNOWLEDGEMENTS

I would like to thank Dr. Peters, Dr. Wilkes, and Dr. Kawamura for all their support and encouragement. Special thanks are due to Dr. Peters for believing in me and advising me throughout this period of much learning. Thanks also to Flo Fottrell, for all her help throughout the years; she has been invaluable during my time at Vanderbilt University.

Thanks to all the CIS team, for all your help, instruction, and advice. I was very happy to work with all of you. Especially to Li Sun, whose direction proved invaluable to my work.

I thank the Electrical Engineering Department and NASA-JSC for their financial support.

Finally, I want to thank my family for all their support. In particular, I want to thank my parents, who worked so hard and sacrificed so much for me to be here today. It would not have been possible if they had not believed in me and supported me to come here. I am eternally grateful for all you represent to me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

From birth, humans make full use of their senses to learn about their environment. Visual, auditory, olfactory, gustatory, and haptic sensations are invaluable entry points for babies to learn about the world. Infants focus in an object or a being and analyze it; they study its shape, color, pattern, and texture; they use touch to sense temperature and texture; smell to know the aroma; and taste to flavor [Lamb 2002]. Furthermore, humans integrate multiple sensory features at a time to dynamically and fully learn about their environment [Pfeifer 1997].

An observe-grasp-reach behavior is typically demonstrated by human babies that are beginning to acquaint themselves with their surroundings. Similarly, when a baby hears a sound, it immediately looks at the corresponding area (perhaps to identify the source) [Irie 1995]. This is exemplary of a set of behaviors that empowers newborns to familiarize themselves with the outside world and to learn the effects of their own actions.

Problem Statement

One goal of research in humanoid robotics is to produce systems that can interact autonomously with people to perform useful tasks. That goal inspired the work reported herein. The objective of this work was to implement fundamental sensory-coupled actions on the Vanderbilt Humanoid, ISAC (ISAC is an anthropomorphic robot found in

the Cognitive Robotics Laboratory at Vanderbilt University. A more detailed description is presented in chapter two) [Kawamura 1995], to construct tasks from sequences of these actions, and to evaluate the results. Ultimately, the goal of this work is to enable ISAC to acquire new behaviors and to perform new tasks

Sensory Motor Coordination

Robots, like humans are capable to learn and interact with the outside world by making use of sensory information. Evidently, the extent and quality of data present in a robot will vary depending on the quality of its hardware and software resources. Nonetheless, sensory motor coordination is essential for the robot to learn basic behaviors to interact with people and its environment.

This thesis will specialize in two specific sensory inputs: vision and audio. Making use of these two inputs a series of basic behaviors were implemented involving camera-head motions and reach-and-grasp motions.

Goals

The goal of the project is to produce a specific set of coordinated reactions to a specific set of data input from the outside world. Once these basic behaviors are well established further progress can be accomplished. Firstly, a set of more complex sequences of reactions can be coupled to produce basic behaviors. Secondly, once coordinated motion occurs, the robot will be empowered to further learn about its surroundings. For instance, visually the robot would be able to examine the shape, color, and pattern of a target and associate these characteristics to that object. Similarly, with

sound, if an object emits a specific sound, the robot could associate the latter with the object.

Organization of This Paper

The remainder of this paper will examine the hardware and software systems used, sensory motor coordination integration, and future work. Chapter 2, RESEARCH TESTBED, will describe the software architecture and the humanoid robot composition used for this work. Chapter 3, SENSORS, will provide an explanation of the different sensors used to gather data from the surroundings. Chapter 4, MOTION, will discuss the different components of ISAC that perform articulated motion. Chapter 5, RESEARCH INTERESTS, will convey the goals of the thesis. Chapter 6, AGENTS, will present agent functionality and goals. Chapter 7, DEMONSTRATIONS, will detail the different behaviors achieved by the robot. Chapter 8, CONCLUSIONS AND FUTURE WORK will present final thoughts and potential future work corresponding to sensory motor coordination behaviors in the ISAC humanoid robot.

CHAPTER II

RESEARCH TEST BED

The hardware and software resources used for this research are described in this chapter. It is important to understand the underpinnings of both hardware and software to have an appropriate sense of the potential, capabilities, and limitations of the systems. In doing so, more effective work can be accomplished.

The ISAC Robot

The humanoid robot ISAC stands for Intelligent Soft Arm Control. The robot was originally created to assist handicapped people. The goal was for the arm to be compliant to ensure the security of the users around the robot [Kawamura 1995].

The safe behavior of the arm is attributed to its compliant nature. The arm is created by joining pairs of agonist/antagonist artificial muscles [Northrup 2001, pp. 73]. These are created from rubbertuators, which are very much like rubber balloons and operate through gas pressure. At its inception, ISAC consisted of one soft-arm, one camera, and a voice system that included speech and speech recognition.

Later, ISAC was given a second arm and along with it a more anthropomorphic shape. Today, ISAC includes a color stereovision system, two independent pan-tilt units, two microphones, an infrared sensor, two soft arms, two pneumatic hands, and speech recognition and text to speech capabilities.

Currently, the arm employs a non-linear controller [Northrup 2001] inspired on a biological system. The hands are composed of a motored thumb and forefinger, and pneumatic distal fingers. This allows for a better grip in the hand. Additionally, the camera control works through a neural network that provides the fixation point for both cameras [Srikaew 2000, pp. 82-85].

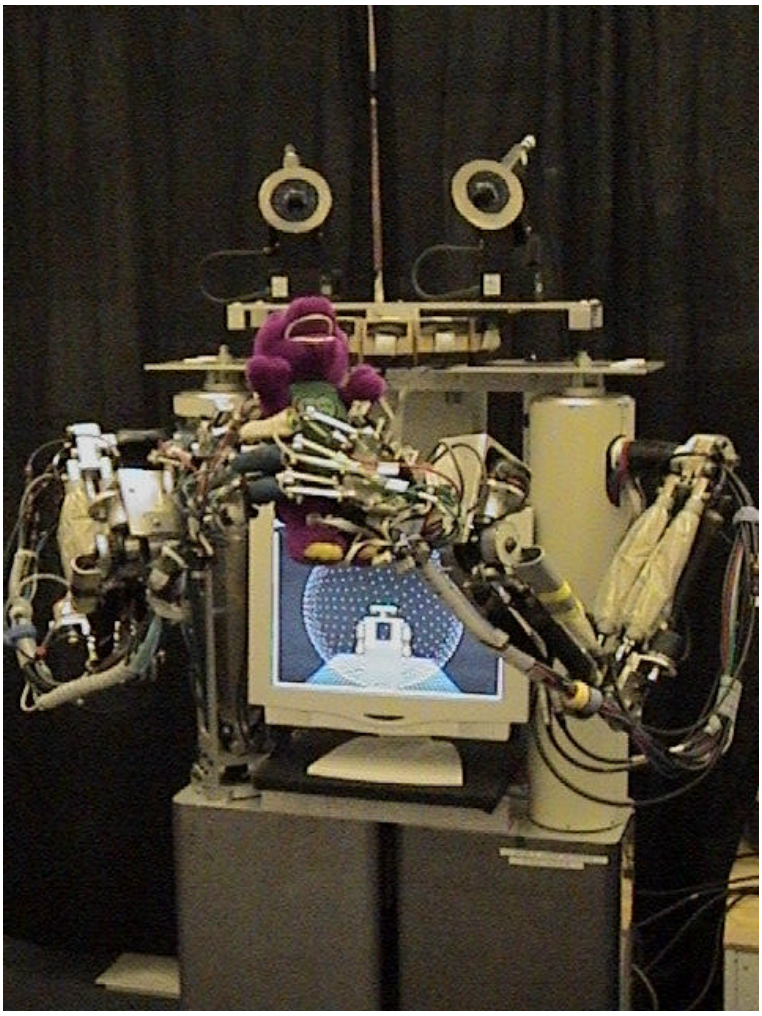

Figure 1. The humanoid robot ISAC.

The IMA Architecture

The intelligent machine architecture encompasses a high and low level of abstraction. At the former level, the architecture consists of a multi-agent system and a multi-agent network. The multi-agent system seeks to model the robot, the tasks it performs, and the environment. The multi-agent network has two functions: first, it controls the robot, and secondly, it is serves as a model for the robot system itself. At the lower level of abstraction, the agent implementation is based on component-object software. Hence, a relationship exists between the distributed agent level and the component-object network. The description borrows heavily from Pack [Pack 1997, 2002].

The higher-level model is named the Robot-Environment model. The agent-based decomposition of this level empowers the architecture to ease the complexity of system integration, particularly by emphasizing encapsulation, reusability, and explicit connections between the agents. The lower level model represents the higher level of abstraction and its relationships by a network of software modules denominated component objects that have an established protocol of communication. Consequently, the IMA is a system of parallel executing software agents formed from simpler and reusable component objects. This architecture is characterized by reusability, extensibility, handling of complexity, parallelism, scalability, reactivity, and robustness. The agent-based system can then serve as a model and a controller for the robot through the representation of lower level explicit software modules that manage the development of software reuse. The Robot-Environment model describes the robot in different sets of resources:

- Physical Resources: arms, head, hand, etc.
- Skill Resources: visual tracking, grasping, etc.
- Behavior Resources: collision avoidance, homing, etc.
- Task Resources: finding objects, moving head, etc.
- Environmental agents: representations of the outside world.

All of these sets are dynamically connected by relationships. Inputs are modified through the computational processes created by these relationships to produce the outputs. In the Agent-Object model, the creation of agents is achieved by combining reusable subcomponents and their parameters. It is then through these relationships and reusable subcomponents that the architecture minimizes its dependence on the internal representations of the system. Figure 2 demonstrates the above description.

The robustness of operation to an evolving system is strong. The network of agents remains unaltered by the changes undertaken in the software modules. The agent level controls the overall architecture yet it remains unaffected by alterations in the mechanisms that manipulate the inputs and outputs. The high level model provides a shell around the implementation level. It isolates the system from changes within an agent. There is great flexibility and operability in that the isolation allows the user to use different mechanisms without changing the overall structure of the system. Additionally, many component mechanisms can be defined at both run-time and design-time allowing for a dynamic configuration. The user is able to refine the performance of the system by executing and correcting faults at run-time.

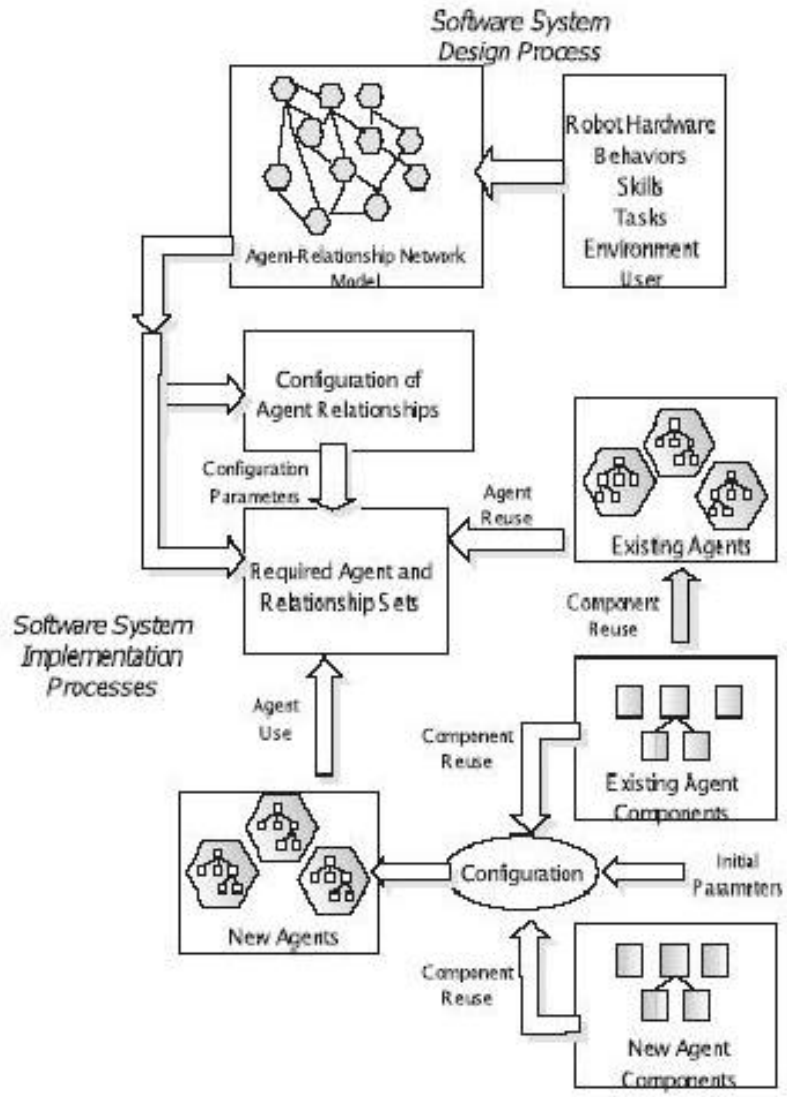

Figure 2. IMA Development Process [Pack 2003, pp. 55].

Robot-Environment Model

Agents can be said to play the role of actors and servers. The former uses other agents as resources and has a thread of execution, and the latter provides resources to other agents as well. It is in the robotics domain that agents provide information to other agents to trigger events. The model of action and communication leads to a different view of design and a more loosely coupled system.

Agents

IMA agents are defined by the following set of properties:

- Autonomous
    - Agents control their own subcomponents.
    - Agents are strongly encapsulated. Hence, their inner mechanisms do not affect other agent functionality.
    - Agents support interaction with other agents through relationships.
- Proactive
    - Agents act locally based on their inner modules, resources, and received data from other agents.
    - Each agent is designed to make a decision or selection based on a state machine or some other process.
- Reactive
    - Agents respond to changes in the environment.
    - Agents run continuously checking inputs, updating their state, and creating outputs.
- Connected
    - Each agent models one concept. So to achieve complex behaviors interaction with many agents is necessary.
    - Connections can transfer simple data like numerical values, or data represented in forms that are more complex.
- Resource Limitation

- All real systems suffer some degree of limitation.
- The system must be able to identify these and adjust accordingly.
- Relationships between agents contain several aspects that allow for this detection and adjustment mechanisms.

One final aspect of IMA is that it is asynchronous. By having asynchronous agents, it is easier to simplify the system at a higher level of abstraction – here synchronization issues become negligible. Each local agent makes its own action decisions based on current data.

Relationships

Relationships capture interactions amongst agents in two specific ways:

- Software interactions: function signatures, sequences, and method calls.
- Structural Interactions: spread activations, motion schemas, etc.

In addition, the communication between agents includes an arbitration mechanism that gives priority to certain agents. IMA relationships represent a variety of arbitrations relationships such as: actuator arbitration, sensory arbitration, goal arbitration, and context arbitration.

Therefore, by the encapsulation of agents the user can evolve individual agents or components in their respective levels of abstraction and further system capability through new relationships without negatively affective system performance.

Agent-Object Model

Agents are concurrently executing modules to achieve a desired goal. They are linked to each other and have at least one thread of execution, some of them having access to hardware. They are also independent decision making modules made out of components that make up the internal representation of the former. Similarly, the agent's overall state is a result of the state of the inner modules and their relationships. An agent's decision could be a command to a hardware resource, a computation, or a communication event.

A model of an IMA agent will now be introduced. This model will provide information on how to implement agents from reusable components. It will allow for explicit representations of software configuration and on-line modification of agent systems. The model consists of four basic factors in providing robot control:

- Agents: provide concurrency, scooping, and decision-making; they are decision objects.
- Relationships: provide interaction protocols and connection ports, they are protocol objects.
- Representations: provide communication, complex information, encapsulation; they are state objects.
- Activities and Actions: provide algorithms, computations, decision process'; they are functional objects.

The model relies in the concept of reusable components to ensure evolutionary growth of the system. The components can be used with different configurations and for different purposes. By simply editing some decision rules, the components can serve

different uses. No new modules need to be built. Integration occurs at a fine level of granularity.

Agents are implemented through a set of components that follow typical design patters in object-oriented systems. They are now presented here and visualized in figure 3.

Agent Managers

- They provide the platform on which the agents are built.
- The main feature highlights a set of on-line interactive tools that allow agent implementation and development.

Agent Engines and Activations

- They conduct the action selection process for individual agents and the invocation of other agents.
- Agent execution occurs in a parallel or sequential ways.
- The Engine is based on a Finite State Machine concept.
- Engines and activities are replaceable and upgradeable.
- The current state machine is set in a hierarchical arrangement.

Agent Representations

  - They serve as invocation mechanisms that perform computations, tasks, and state updates.

Agent Resources

- Resources available to agents include: data repositories, algorithms, and links to the states of other components.
- They can be evaluated, invoked, and updated within an agent.

- Components can be simple and complex.

Agent Component Managers

- They are wrappers for component-objects.
- They handle object persistence and provide a visual representation of the internal state of a component allowing the agents performance to be updated.

Relationship Components

- They encapsulate the connections between agents.
- Connections range from simple data-flow to multi-way data arbitration mechanisms.
- These components are encapsulated relationships that isolate the agents from changes in the relationships amongst themselves.

Code Development Framework

In this section of the IMA description, abstract concepts are presented as a conceptual guide to develop IMA agents. The purpose is to encourage the developer to think in terms of large scale, more reusable components, rather than a simple algorithm or monolithic program. The abstractions of representations, mechanisms, and policies reduce the amount of any "hidden" interactions between components and force them to be explicit.

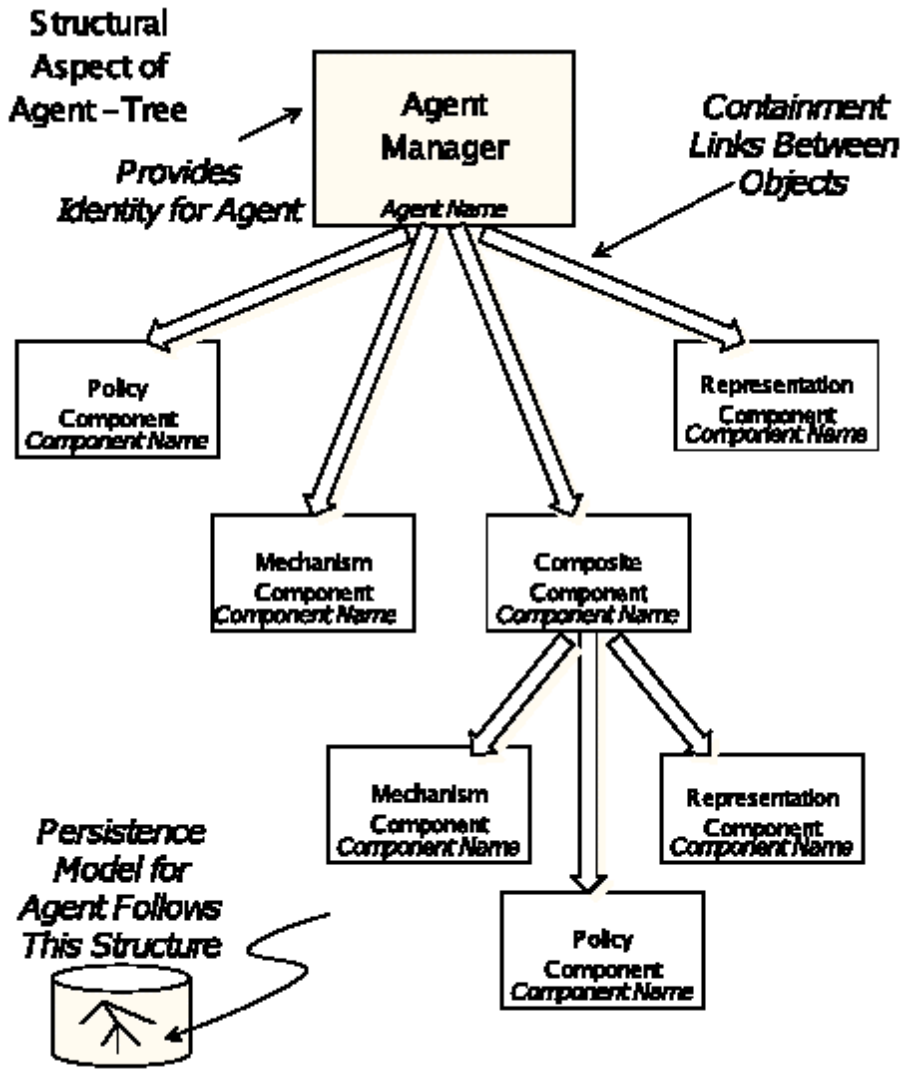

Figure 3. IMA Agent Internal Structure [Pack 2003, pp. 66].

For an architecture to be useful, it needs to make the work of the developer easier, faster, and more productive. With this in mind, a set of base classes written in C++ was created to reduce the effort needed to follow the constraints of the architecture. The code framework created provides a starting point to the agent-object model from a more traditional programming language. These base classes provide most of the common functionality required to create these agents. They typical functionality is now listed:

IMA Component

- It is the generic class for an agent component. It requires minimal naming, persistence, and linking.

Mechanism

- It adds the invocation mechanism feature to the base class. It also provides a standardized way to invoke the mechanism.

Representation

- It adds proxy and source functionality and interfaces that support communication between a sender source and proxy representations.
- Representations encapsulate communication across sources and proxy's.
- Mechanisms can be aggregated because this level of communication is encapsulated.

Policy

- This class has a scheduled thread of execution based in the operating system.
- The class allows sequencing and decision-making processes' to occur.
- They invoke mechanisms in the agent state or external agents in which they are found.

Agent Manager

- It provides a platform for agent timing services, persistence management, and event dispatching.
- The manager also takes care of the agent locator and the agent shell. This allows components to bridge the gap with the outside world.

Personal Review

Through my personal experience with IMA I have discovered this architecture to be of great utility in working with robotic systems that demand a multi-agent system that is strongly encapsulated, scalable, modular, and reusable.

The key to use the full of potential of the IMA the developer must be strongly familiarized with the inner workings of the architecture, its user interface, its state machine structure, and its communication protocol. Those that have used the IMA architecture to develop have criticized the architecture and labeled it as complex and slow. In my opinion, complexity and speed issues can be solved with a good understanding of the system. Evidently, there are down sides to the architecture and I will provide my view of these later on.

If the developer has a clear conceptual model of the process to be implemented, IMAs strong encapsulation and scalability provide ample flexibility to design a precise system. The robustness and efficiency of the system will then be a reflection of the quality of the programmed algorithms. Poor programming skills will be evident at the Robot-Environment level. Similarly, for the system to be quick to react and respond, proper parameters need to be established. Speed can be influenced by several factors, such as: the length and complexity of a calculation, the specified period of a given state in a state machine, and the number of event calls within a state. After much experimentation, many of the challenges described above were overcome. This most demanding challenge of the system is learning and becoming familiarized with it. Acquainting myself with IMA took many hours of hard work.

A list of recommendations for improvement would include: a more user-friendly GUI; more reliable communication amongst agents - at times agents do not recognize certain calls; finally, the state machine is designed to operate in a sequential manner – and limits the functionality of the system. A distributed-parallel organization for the state machine would allow agents to run simultaneously and thus have multiple processes running in the humanoid. An improved version of IMA could be very useful in our continued efforts to research humanoid robotics.

CHAPTER III

SENSORS

For the purposes of this research, two main sensory inputs were sought – vision and audio. Two stereoscopic color cameras and two high fidelity microphones act as entry points from the environment to the robots data structures. Audio signals serve to direct the robots attention to an area of activity. Visual information goes further and indicates the specific location (given by pan and tilt angles) of a given target.

Vision

The primary sensory modality of humans is vision. Similarly, for humanoid robots, vision represents the most important sensory modality as well. Vision recognition of the environment is necessary for the robot to analyze its environment and perform intelligent motion behaviors. Robotic vision has been a very challenging field to scientists. The goal of this thesis is to direct the attention of the camera-head towards the object of interest through pan and tilt commands. A description of the hardware and software tools will follow. The latter will include an explanation of simple yet useful tools used to perform color segmentation and motion detection.

Cameras

Two Sony XC999 cigar cameras were used for the camera-head of to achieve stereoscopic vision. The camera is an ultra-compact and lightweight, one piece, cigar

camera with 0.5 inches of a colored CCD array. A CCD array is described as a charge-coupled device array. The latter contains light sensitive diodes that sweep across an image and generate a series of digital signals that are converted to pixel values. CCD cameras are characterized for low noise to signal ratios. The camera also features hyperHAD technology, which outputs pictures of high digital quality with a power requirement of 12VDC. Additionally, RGB signals and illumination levels can be adjusted externally [Sony XCC 2004].

Color Models

Before delving into the theoretical underpinnings of color vision, it is important to clarify the goal of the work done in image processing. A simple but efficient image tools was designed by Barile [1997] for the humanoid robot ISAC. It was desirable to direct the focus of attention of the robot to specific objects in the environment. To do this, objects were detected quickly and robustly by using color segmentation. Incoming images from the cameras are compared against predetermined color models to determine if there is a match. Post-processing on the segmented blobs leads to the detection of contiguous regions of similar chromaticity. The center of mass is then calculated and chosen as the focus of attention. Once a point has been chosen, camera movement algorithms can be chosen to direct the camera head to the appropriate location in space. Color images are dependent on different chromaticity and brightness variables. Thus, color segmentation techniques must account for two factors to achieve appropriate results.

Colors can be described in a variety of different spaces. Each color space model exploits different properties of the visual field, and thus their uses vary according to the needs of the user. A brief exposition of the color space models will be presented next:

RGB Color Space

- This model expresses color as a linear additive relationship between the three primary colors known as red, green, and blue, and white light.
- The three primary colors are based on a Cartesian coordinate system.
- Each color can be determined with a vector representation of these colors.
- There are a few disadvantages to the model:
    - The spectral power distribution of the RGB model varies depending on the type of hardware used. The spectral power distribution can be described as the characterization of light by assigning a measure of power at every wavelength in the visible spectrum [Spectral Power Distribution 2004].
    - The RGB space provides a single measurement that contains information for both chromaticity and illumination; this lack of distinction makes this model inadequate for image processing.

In view of this problem, the CIE (Commission Internationale de L'Eclairage) established a new color model with a new set of standard primaries known as the CIE 1931.

CIE XYZ

- This model separates both luminance and chromaticity into two components:
    - Luminance component = Y.

- Two chromaticity components = X and Z.

- This is a standardized model, which renders it independent from hardware-based materials.
- The spectral power distributions of the primaries X, Y, and Z, were established at 5nm intervals between the wavelengths of 360nm and 830nm.
- RGB to XYZ conversions are possible:

Table 1. RGB to XYZ color space conversions [Barile 1997, pp. 9].

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & .180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} * \begin{bmatrix} R_{709} \\ G_{709} \\ B_{709} \end{bmatrix}$$

- The luminance value of the CIE space is intended to reflect the luminous efficiency of the human eye.
- CIE reveals perceptual non-uniformity. In other words, changes in the wavelength of the colors do not result in an equal change in the color perceived.

Next, a description of normalized color spaces is presented. Normalized color spaces help to reduce color representations from three to two dimensions and helps to recover the true chromaticity in RGB or XYZ models regardless of the intensity of the illuminant.

Normalized Color Spaces

- In general, there are two methods of normalization:
  - The L^1 normalization process allows scaled RGB values to lie on a plane where their maximum value is equal to one.

$$r = \frac{R}{R+G+B} \tag{3.1}$$

$$g = \frac{G}{R+G+B} \tag{3.2}$$

$$b = \frac{B}{R+G+B} \tag{3.3}$$

$$\begin{aligned} r+g+b &= 1 \\ b &= 1-(r+g) \end{aligned} \tag{3.4}$$

- This implies that only two of the three values are needed to know the complete model. Commonly, red and green values are used, since the human eye is not significantly influenced by the blue color.
- The Euclidean distance between two normalized points depends on two angles. One found in between two adjacent lines - q, and the other one found between the first line and the x-axis, q_1. See figure 4.

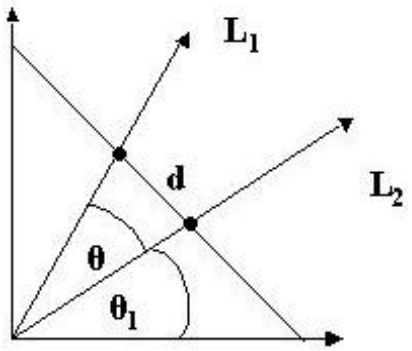

Figure 4. L^1 norm model [Barile 1997, pp. 10]

- The L^2 normalization differs from the first one in that the values here are scaled onto a spherical line rather than linear model.
    - The Euclidean distance is a function of the angle between the lines. As opposed to the first technique, the Euclidean distance is a function of the angle made by the two lines that meet the points in space. This makes the L^2 norm model useful for color segmentation methods that depend on color differences. See figure 5.

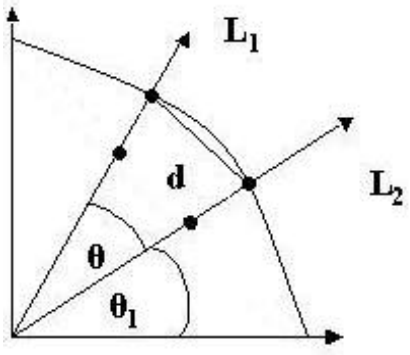

Figure 5. L^2 norm model [Barile 1997, pp. 10]

CIE $L^* u^* v^*$

- The $L^* u^* v^*$ color space was designed to overcome problems of perceptual non-uniformity present in the other color spaces.
- In this model, differences in wavelength size are equally perceivable by changes in color.
- Light is represented in terms of luminosity – L, and chromaticity – u and v.
- The topology of the $L^* u^* v^*$ color space is illustrated in the figure 6.

- L*u*v* values can be calculated from non-linear transformations from the CIE XYZ space. It also depends on a white point (X_n, Y_n, Z_n):

$$L^* = \{\frac{116(a)^{1/3} - 16}{903.3a} \text{If a > 0.01} \tag{3.5}$$

Otherwise,

$$u^* = 13L^*(u' - u'_n) \tag{3.6}$$

Where,

$$a = \frac{Y}{Y_n} \tag{3.7}$$

$$U' = \frac{4X}{X + 15Y + 3Z} \quad v' = \frac{9Y}{X + 15Y + 3Z} \tag{3.8}$$

$$U_n' = \frac{4X}{X_n + 15Y_n + 3Z_n} \quad v'_n = \frac{9Y}{X_n + 15Y_n + 3Z_n} \tag{3.9}$$

Similarly, a separate related to color must be described. It is termed color constancy, and it describes the adaptive behavior seen in humans, whereby the colors perceived in environments that are illuminated by difference sources still look the same way. Actually, the wavelengths of the reflected colors from objects under different illuminants do change, but the brain uses an adaptation process that allows the human to perceive the same chromaticity. This behavior is of much value to robotic systems. It provides robustness to changes in the source of the illuminant, especially for those systems that do image processing through color techniques.

The outside world to the robot is dynamic and ever changing. Different visual effects are always present and the visual system of the robot needs to overcome the

unstructured nature of the environment. The algorithm that was implemented to produce color models for given image data was designed with an unstructured environment in mind. A simple reduction technique is used; it minimizes the effects of changes in brightness. It does so by converting RGB images to the XYZ color model.

Color Segmentation

Color segmentation is a technique used in image processing to reduce the amount of information to analyze from the environment. In using this technique, one must make sure that the segmented regions represent a specific feature in the environment based on a standard criterion. Segmenting by color can involve different techniques such as: histogram modeling techniques, clustering, and neighborhood algorithms (region growing and split-and-merge techniques). The color segmenting software implemented for ISAC utilized histogram-modeling techniques.

Commonly, histogram modeling is performed on *a priori* data models from which statistical models can be obtained. For ISAC the model is based on probability ellipses, which perform statistical analyses on single colored objects.

The algorithm requires a predetermined segmented color to perform the statistical model on the data. Hence, the color segmentation is done beforehand. A choice of color space can be made by the developer. RGB, RG-normalized, and L*u*v* models are available. Finally, the algorithm creates a statistical model on the data and stores it in a knowledge base for later use.

When an image is classified, every pixel value is analyzed in the image to determine if it fits the model. If true, then that pixel location is segmented. Image

preprocessing (downsampling and blurring) is available to the user to reduce the amount of information and noise. Post processing is also done in the form of a morphological opening – that is an erosion followed by a dilation with a 3x3 pixel cross. The operation aids in overcoming effects from spurious pixels from the background, highly textured surfaces and camera noise.

Probability Ellipse Model

The color histogram of a green disc under fluorescent light in the normalized rg and L*u*v* color spaces is illustrated in figure 6.

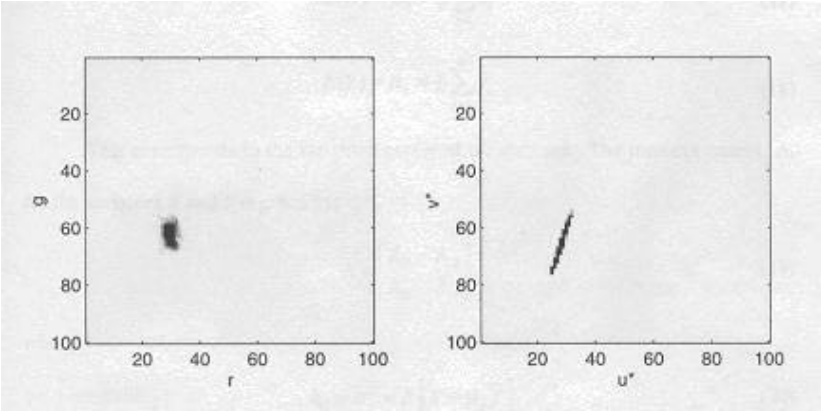

Figure 6. Histograms under fluorescent lighting: (left) Normalized rg color space, (right) L*u*v* color space [Barile 1997, pp. 23].

It is evident that for each color space, the presence of the pixel values is clustered around a specific region in space. This property can be used by modeling the data as a two-dimensional joint normal distribution, or probability ellipse. Given N samples from two random variables X and Y, the expected values are given by:

$$E(X) = m_x = \frac{1}{n}\sum_{i=1}^{N} x_i \qquad (3.10)$$

$$E(Y) = m_y = \frac{1}{n}\sum_{i=1}^{N} y_i \tag{3.11}$$

The expected values correspond to the mean value of the data set. The moment matrix, Λ , for the variables X and Y is given by:

$$\Lambda = \begin{pmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{pmatrix} \tag{3.12}$$

Where,

$$l_{11} = s_x^2 = E\{(X - m_x)^2\} \tag{3.13}$$

$$l_{22} = s_y^2 = E\{(Y - m_y)^2\} \tag{3.14}$$

$$l_{12} = l_{21} = Cov(X,Y) = E\{(X - m_x)(Y - m_y)\} \tag{3.15}$$

Where, s_x^2 and s_y^2 are the variances of X and Y. There is also a correlation coefficient, r , is a measure of how closely related the two variables are and is defined by:

$$r = \frac{l_{12}}{s_x s_y} \tag{3.16}$$

Where, s_x and s_y are the standard deviations of X and Y respectively.

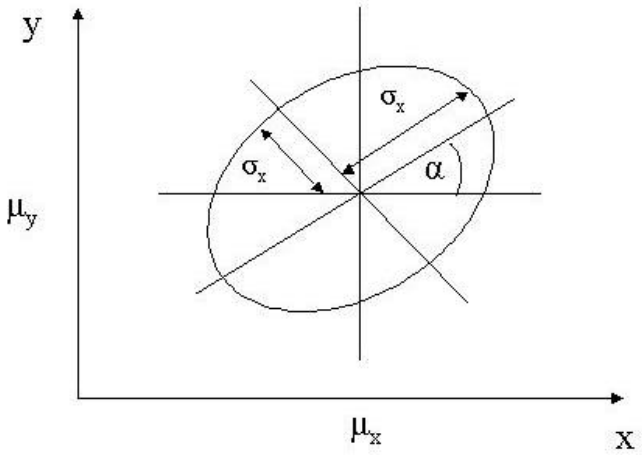

Figure 7. Geometry of the probability ellipse [Barile 1997, pp. 25].

The parameters $m_x, m_y, s_x, s_y,$ and r, are used to find the probability ellipse with uniform distribution inside and zero distribution outside. The equation for the ellipse is:

$$\frac{1}{(1-r^2)}\left[\frac{(x-m_x)^2}{s_x^2} - 2r\frac{(x-m_x)(y-m_y)}{s_x s_y} + \frac{(y-m_y)^2}{s_y^2}\right] = l^2 \qquad (3.17)$$

In figure 7, the center of the ellipse is found at (m_x, m_y). The parameter ? is the number of standard deviations captured by the ellipse. The long side of the ellipse is equivalent to $l\,s_x$, while the short side is equal to $l\,s_y$. The angle of rotation of the ellipse about the center, a is defined by:

$$a = \tan^{-1}\left[\frac{1}{2rs_x s_y}\left[s_y^2 - s_x^2 \pm \sqrt{(2rs_x s_y)^2 + (s_y^2 - s_x^2)^2}\right]\right] \qquad (3.18)$$

Hence, we can model the color of interest through this statistical procedure in a two dimensional color space.

Pixel classification checks to see if the pixel value falls within the ellipse's boundary. The user would choose a value for the standard deviation, ?, and then do a pixelwise computation to find the pixel value:

$$a = \frac{1}{l^2(1-r^2)}\left[\frac{(x-m_x)^2}{s_x^2} - 2r\frac{(x-m_x)(y-m_y)}{s_x s_y} + \frac{(y-m_y)^2}{s_y^2}\right] \qquad (3.19)$$

If, $a \le 1$, the pixel is part of the ellipse and it is segmented.

Focus of Attention

Once all the segmentation and post processing has been done, the algorithm finds the center of mass of the set of all foreground pixels. This is a simple procedure and is defined as:

$$F_x = \{x \mid B(x, y) = 1\} \tag{3.20}$$

$$F_y = \{y \mid B(x, y) = 1\} \tag{3.21}$$

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} F_{xi} \tag{3.22}$$

$$\bar{y} = \frac{1}{N} \sum_{i=1}^{N} F_{yi} \tag{3.23}$$

Where, N is the total number of foreground pixels. This is a simple and quick method to find the center of mass. Although, it makes the assumption that only one object is being analyzed. If more than one object is segmented, the FOA will be somewhere between the objects.

Motion Detection

The motion detection algorithm is a simple frame-differencing algorithm. Consecutive image frames are used to subtract the luminance values for each image. By using a predetermined value, the differences in luminance values are compared to the threshold. If the result is greater than the threshold, it represents a pixel that did not change in value, and thus is part of the moving object. All such pixels are segmented and later used to determine the center of mass of the blob. Preprocessing techniques like blurring and down sampling are used to reduce the amount of noise in the image. It is important to note that this is a simple and not very robust technique when coupled with the pan-tilt units. As soon as the head unit moves, noise is introduced in the image. However, the threshold value along with the specific type of motion allow for a fair tracking behavior of the camera head.

Audio

In this section, an introduction to auditory hardware and software modules will be purported. The bulk of this work specialized in finding an auditory cue as sensory input for the robot to detect the presence of activity in the environment.

Microphones

The MXL microphone is characterized by its pre-amp circuitry and a balanced transistor output for maximum frequency response. It has high sensitivity and low distortion recording capabilities. The frequency range captured by the microphone is between 20 kHz and 30 kHz. The microphone needs 48V of phantom power and it is known for its performance and reliability. The microphone is shown in the image below.

Figure 8. MXL Microphone [MXL 2004]

Sonic Localization

Sonic localization in humans is possible through a binaural mechanism. The dual sound receptors provides audio cues in the horizontal plane and in the vertical plane [Irie 1995, pp. 15]. For ISAC, a simple but efficient sonic localization algorithm was sought.

Using two microphones, the direction of sound can be easily computed in the horizontal plane.

The algorithm was based on the premise of comparing the sound energy ratio between the right and the left audio channels for selected space locations. A sound intensity envelope was collected and filtered for both channels. The sum of the squares for each signal was computed and used as a measure of energy. The ratio between both channels was compared at eleven different locations:

Table 2. Sonic Partitioning.

-90	-60	-45	-30	-15	0	15	30	45	60	90

The values of the ratios give clues as to the direction of the signal. The values are used as a reference for future measurements and used to determine the location of incoming signals. The standard ratio values can be calibrated at any time, thus providing an easy was to adjust the system to different noise conditions.

Once an incoming signal is received, the ratio of both channels is computed and by means of linear interpolation, a single angle interval is selected as the direction of the sound cue. The latter is presented as a pan angle that can be utilized by other agents in the system.

CHAPTER IV

MOTION

Once the robot possesses knowledge about its environment, behavioral responses can take place. This is similar to psycho-physiological tests where behavioral responses to low-level inputs are examined. Hence, when the appropriate image processing techniques have taken place, a spatial location is retrieved and passed to the pan-tilt unit to produce the appropriate motion. Furthermore, when the pan-tilt unit fixates on the desired object a Cartesian location in space can be calculated from the angles of the cameras. This position is used to command the arm to articulate its motion towards the gazed target and perform a reach-and-grasp behavior.

Eyes

The visual system of the robot is implemented through both the hardware and the software. The hardware cameras allow the robot catch images of the outside world. This single sensory input is the most significant of all sensory input and thus provides the most information. The images then needed to be analyzed as was described in the sensors section. Yet, for the information to be of true value to the robot, it needs to be corresponded with meaningful movement on the robots part.

Pan-Tilt Unit

The pan-tilt unit is a high-speed and accurate positioning camera, reaching speeds of up to 300 degrees/second. Two separate and independent pan-tilt units are connected to a 1400MHz computer via two RS-232 connections.

Figure 9. Pan-Tilt Unit [Direct Perception 2004]

Head Controller

The camera head controller is the agent component that deals with the pan-tilt unit motion. The controller should allow the following properties to take place:

- The pan and tilt motions for each individual camera should be precise and available for all agents in the system.
- Each pan and tilt motion should be independently controlled.
- Velocity and acceleration should be controllable.

The pan-tilt units used on ISAC react quickly and precisely to inputs from the head controller. The left pan and tilt motions are controlled by the first RS-232 port, while the right pan and tilt motions are controlled by the second port. In both units, the

velocity and acceleration are controlled. The information contained within this section was largely produced by Atit Srikaew [Srikaew 2000].

Saccades

Saccades are typical eye movements that help focus the object of attention unto the fovea. Similarly, the purpose of a saccadic motion in a visual system is to move the gaze of the controller to a point inside the fovea. The saccade function in the system was created by making use of two modules: the saccade map trainer and the saccade command generator. The advantage for using a neural net as a mapping trainer is that no camera calibration is needed and it can be done in a fast and accurate way. It is important to recognize the fact that the original neural network was implemented for a different head configuration. At the time of the hardware modification, it was deemed acceptable to keep the output of the network for the new configuration due to the similarities between the old and the new structure. Training with the new head will be part of future work. Nonetheless, the description of the current model is presented below.

The map trainer provides an adequate transformation for the saccade command generator to issue accurate commands to the pan-tilt unit. The training is done off-line by using a back-propagation learning algorithm. Each pan-tilt unit is trained individually. The inputs to the neural net are the ?x and ?y position of the gazed target, whilst the output nodes are those corresponding to the pan and tilt motions respectively. A hidden middle layer that contains 25 hidden neurodes is used and is governed by a bipolar hyperbolic tangent sigmoid function. See figure 10 below for a diagrammatic representation of the neural net.

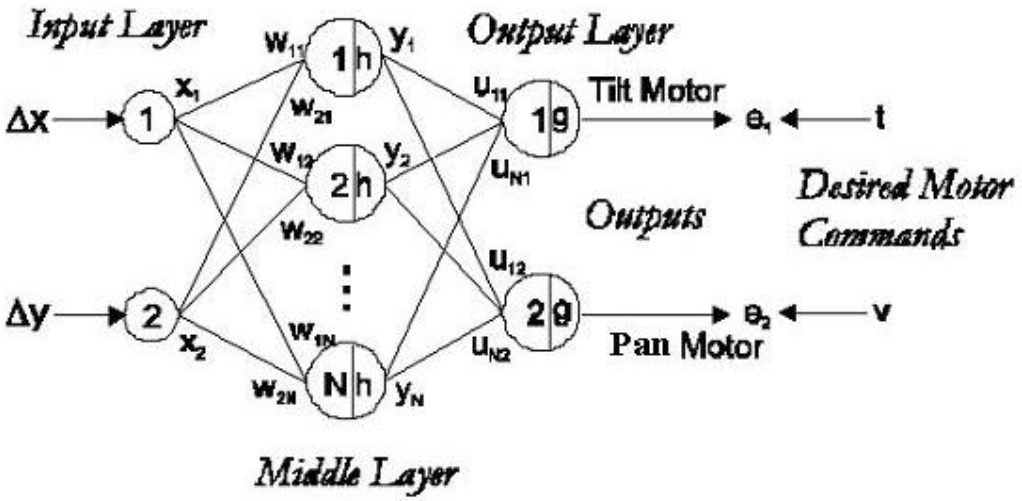

Figure 10. Feedforward Neural Network for Saccade Training [Srikaew 2000, pp. X; this figure has been modified to reflect updates in the system].

By using sample input-target pairs shown in figure 11a and a sum-square error of 0.02, the resulting map shown in figure 11b was obtained:

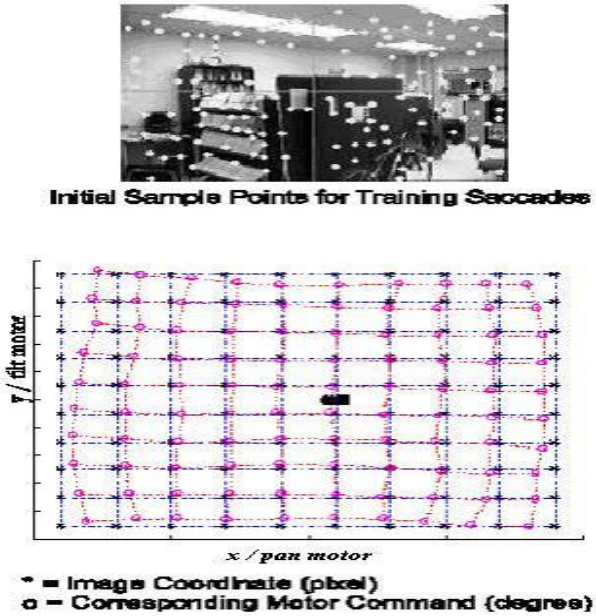

Figure 11a and 11b. Saccade Map Training [Srikaew 2000, pp. 102].

The saccade command generator uses the map to produce motor commands corresponding to target-position inputs. A standard feed-forward network is used to calculate the output values, which are sent to the Eye Motion Center to control the camera head. See figure 12.

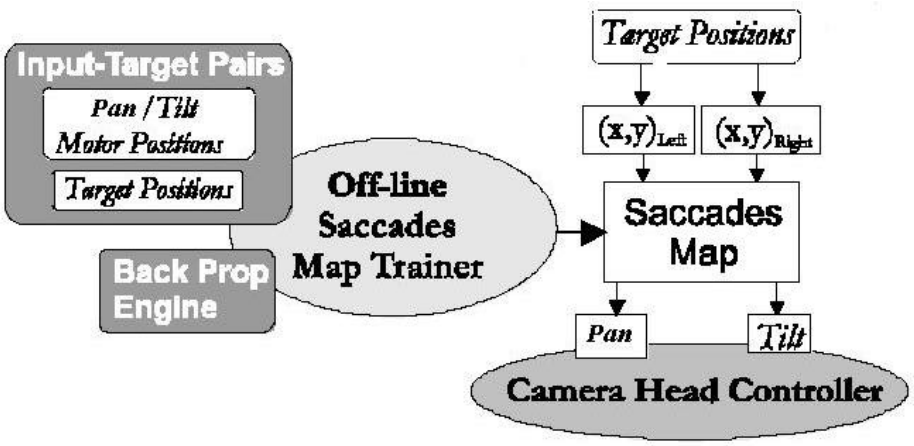

Figure 12. Implementation of Saccade Control [Srikaew 2000, pp. X; this figure has been modified to reflect updates in the system].

Post saccade processing was performed to correct the neural weights in the network to minimize the error of the transformation.

Smooth Pursuit

Smooth pursuits are designed to keep a target in the fovea. The eye movement utilizes two parameters to perform its motion: target position and target velocity. The latter is used to predict the future position of the camera and allow for a smoother trajectory in the camera head. See figure 13.

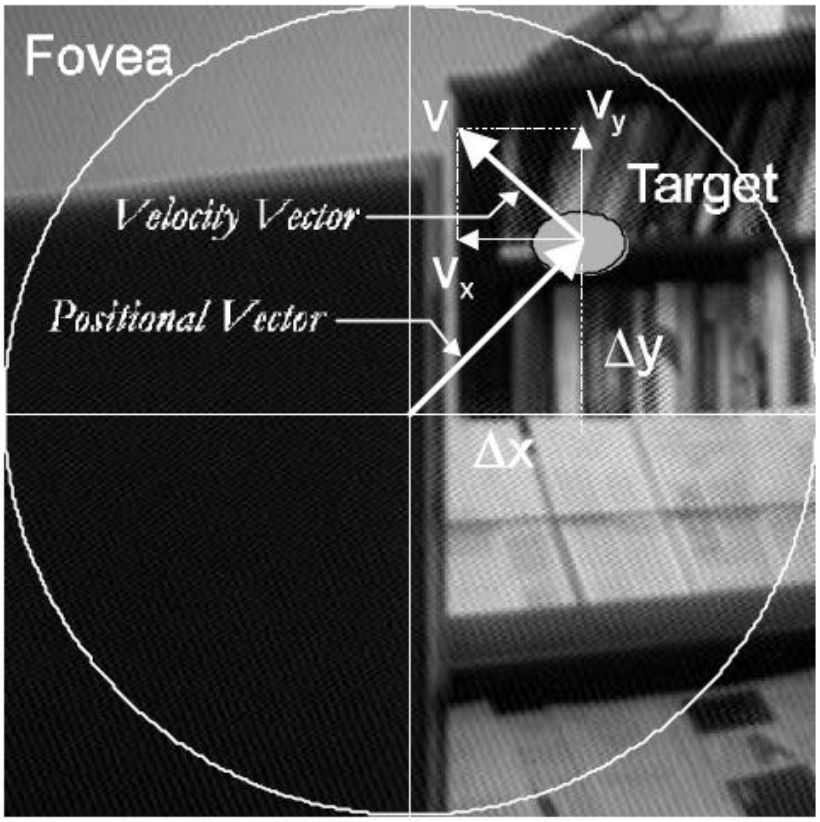

Figure 13: Positional vector and velocity vector used for smooth pursuit control [Srikaew 2000, pp. 88].

Two concentric regions of different radii are defined as: the fovea with a radius of *F* pixels, and the dead zone with a radius of *D* pixels, where, $F > D$. See figure 14. Smooth pursuit's also known as proportional tracking motions occurs when the target is located outside the dead zone but inside the fovea. If at any time the target exits the fovea, a saccade is used to reach the target.

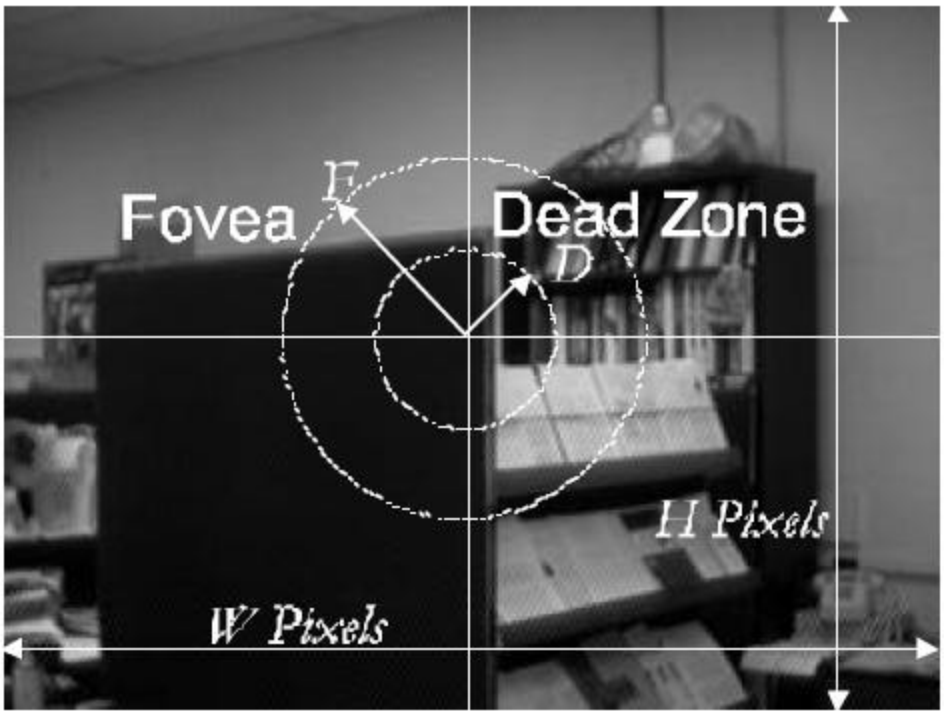

Figure 14. Definition of Fovea and Dead Zone Area in Image Plane [Srikaew 2000, pp. 86]

The positional vector is the distance of the target from the center of the fovea and is defined as:

$$\vec{p} = (p_x, p_y) = (\Delta x, \Delta y) \tag{4.1}$$

$$\left|\vec{P}\right| = \sqrt{\Delta x^2 + \Delta y^2} \tag{4.2}$$

The velocity is described equally:

$$\vec{v} = (v_x, v_y) \tag{4.3}$$

$$\left|\vec{v}\right| = \sqrt{v_x{}^2 + v_y{}^2} \tag{4.4}$$

Where, the units are in pixels per time unit.

It follows that the motor commands $m_{PL}, m_{PR}, m_{TL}, m_{TR}$, can be calculated by making use of the distance of the target and constant gains $k_{PL}, k_{PR}, k_{TL}, k_{TR}$, for the left pan, right pan, left tilt, and right tilt motors.

$$m_{PL} = k_{PL} * \Delta x_L \tag{4.5}$$

$$m_{PR} = k_{PR} * \Delta x_R \tag{4.6}$$

$$m_{TL} = k_{TL} * \Delta y_L \tag{4.7}$$

$$m_{TR} = k_{TR} * \Delta y_R \tag{4.8}$$

Where, the *L* and *R* subscripts describe the left and right images respectively.

To predict target position, let $?t$ be the time interval between consecutive image frames, let $\dot{v}$ be the velocity of the target at time *t*, so the position of the target at the next frame $t + ?t$ be:

$$\Delta x_{t+\Delta t} = x_t + v_x * \Delta t \tag{4.9}$$

$$\Delta y_{t+\Delta t} = y_t + v_y * \Delta t \tag{4.10}$$

This produces a smooth motion in the camera head when it tracks a given target.

The camera head controller is an open loop controller and receives no feedback. Object trajectories vary dynamically with the environment making it hard to estimate the exact trajectory. Overshoots in the camera head may occur; to lessen this effect a low-pass filter is used to reduce the overshoots. See figure 15.

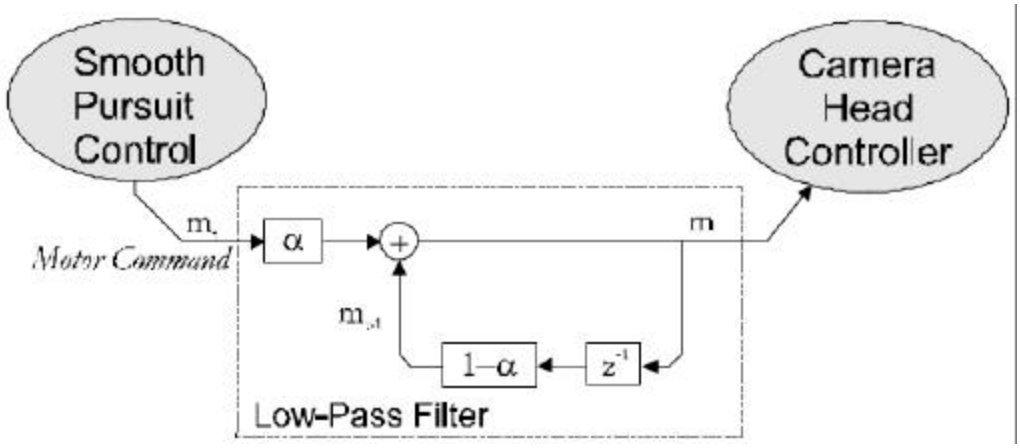

Figure 15. Low pass filter for smooth pursuit [Srikaew 2000, pp. 105].

The filter is applied to each motor command before it is sent to the pan-tilt unit. Let m_t be the current motor command, and m_{t-1} be the previous motor command. The low-pass filter signal, m, is determined by:

$$m = a \times m_t + (1 - a) \times m_{t-1} \tag{4.11}$$

Where, a is a filter constant and $0 \le a \le 1$. If $a = 1$, $m = m_t$. Hence, the filter's effect is negligible. The value of a was determined empirically to obtain the best performance in the camera head.

System Integration

The visual system combines all the modules contained vision sensing, and eye motion. The images provided by the cameras are captured by a device that encapsulates all frame grabber functionalities. Then, objects can be extracted from the environment by means of color segmentation of motion detection. The center of mass of the segmented blobs is calculated and used to provide target motion information and target position information. The former yields a velocity signal later used by the smooth pursuit module,

while the latter yields a target position used by the saccade module. Both eye movements, experience a slight delay that accounts for the movement of the camera. Yet, data streams are passed as quickly as possible to keep the camera head on target. Saccades and smooth pursuit's behave similarly to each other. These behaviors can be executed independently or in conjunction.

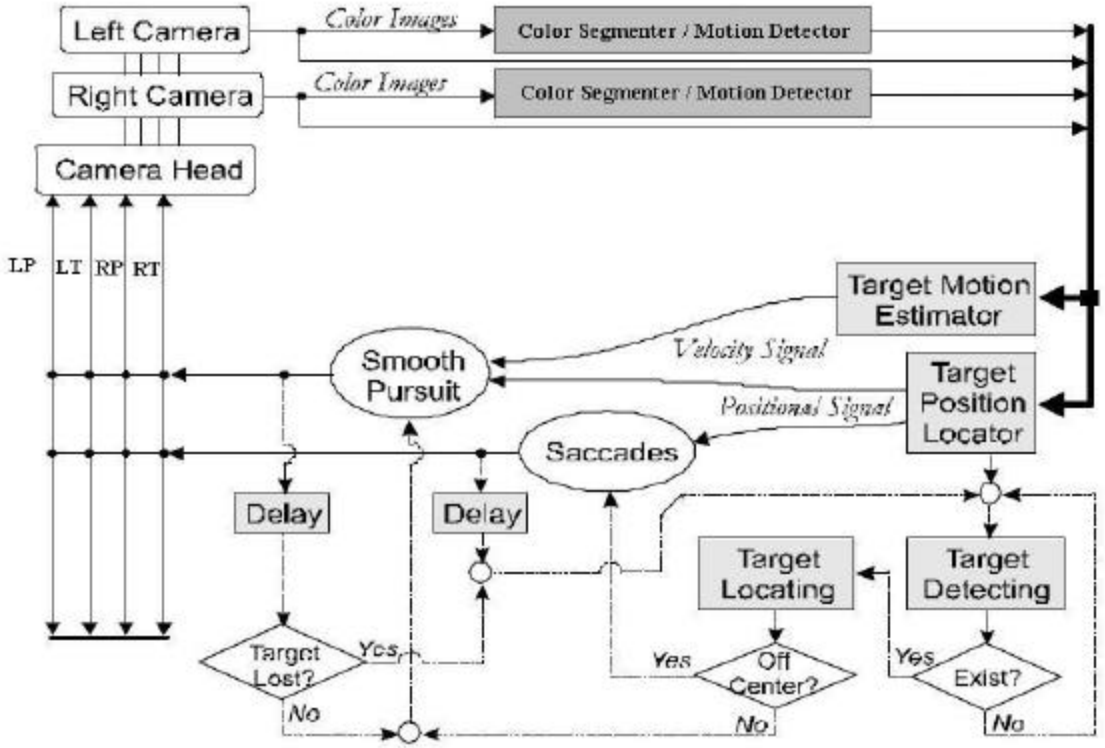

Figure 16. System Integration for Visual system [Srikaew 2000, pp. 116; this figure has been modified to reflect updates in the system].

Pneumatic Arms

The humanoid robot, ISAC, is actuated by pneumatic "artificial muscles". These were first developed by J. L. McKibben in the 1950's as orthotics for polio patients [Klute, 1999]. Klute and Hannaford [1998] describe them as actuators made from an inflatable, tubular inner bladder sheathed with a nylon double helix weave that shortens

lengthwise when expanded radially [1998]. Those two main components are clamped with fittings at both ends, one of which contains an air intake. The nylon sheath holds constant the volume of the gas within the rubber tube. Therefore, as they are inflated, the actuators contract along the axis of the tube. Similarly, as they deflate they expand along the axis. If one end is fixed, the other will move a load in an approximately linear fashion [Daerden 2002]. The arms exhibit compliance as a direct result of two factors: a pneumatic actuator operates on the basis of gas compressibility, and its inner bladder is elastic. Even if the gas pressure remains unchanged, an applied force that changes the length produces a spring-like behavior in the rubber material of the bladder which enhances the compliance of the actuator beyond the compressibility of the gas. Because of their constituent materials, McKibben air muscles are lightweight and have a characteristically high force to weight ratio. Other significant features include direct hands-on connections, easy replacement, and safety due to their natural compliance [Klute 1998, Daerden 2002].

At its inception, ISAC used "Rubbertuators" — McKibben air muscles produced by the Bridgestone Corporation. The Rubbertuators have since been replaced with UK's Shadow Robot Company air muscles. These new arms produce less hysteresis and require less power consumption than the original Bridgestone actuators.

Figure 17. Shadow's Rubbertuators [Shadow Air Muscles 2004]

ISACs arms have six degrees of freedom: a base that rotates about the vertical axis, a shoulder that rotates about the horizontal axis; and an elbow and wrist that rotate both around the x- and y- axes.

Each joint has a pair of air muscles that act as opposing muscles. They are defined as agonist and antagonist muscles – they need to be coupled to produce motion in two directions. As one actuator pushes a load, the other one will act to stop it, by balancing the pressure in the arms the desired position can be achieved. The control loop and model will be explained next.

Control

This section will include a complete description of the control mechanisms for the soft arms of the humanoid robot ISAC. Hardware details will be discussed first, secondly they the control loop paradigm is presented, followed by the controller model and implementation, and finalized by the arbitration loop.

Hardware Details

The air flow necessary for the artificial arms to work runs through a compressor and an air dryer to ensure the purity of the air - rubbertuators and valves are susceptible to impurities in the air [Alford 1999].

Physical angles in the arm are computed by converting encoder angles by the following equation:

$$PhysicalAngle = \frac{(RawEncoder - EncoderOffset)}{EncoderGain} \quad (4.12)$$

Most angles in the system deal with logical angles as opposed to physical angles. The joint space angles and the physical angles differ by virtue of the geometry of the robot. The following equations are used to map between these two forms:

$$L1 = p1 \quad (4.13)$$

$$L2 = p2 \quad (4.14)$$

$$L3 = 0.5(p3 + p4) \quad (4.15)$$

$$L4 = p3 - p4 \quad (4.16)$$

$$L5 = 0.5(p5 + p6) \quad (4.17)$$

$$L6 = p5 - p6 \quad (4.18)$$

The equations can be rearranged to solve for the physical angles:

$$p3 = L3 + 0.5 * L4 \quad (4.19)$$

$$p4 = L3 - 0.5 * L4 \quad (4.20)$$

$$p5 = L5 + 0.5 * L6 \quad (4.21)$$

$$p6 = L5 - 0.5 * L6 \quad (4.22)$$

Forward and inverse kinematics can be used to move the arm to desired positions. The control loop is described next.

Control Loop

The control loop is explained in the CIS Technical report [1999]. Figure 18 shows these steps that are delineated below:

- Position commands are introduced via the “Desired Logical Angles” component.

- The commands are filtered with a first order IIR low-pass digital filter.
- The filtered angles are converted by the sampler mechanism to "Desired Physical Angles."
- The desired and actual physical angles are used to compute the pressure output.
- The sampler then drives the servo valves which control the pressure in the artificial muscles.

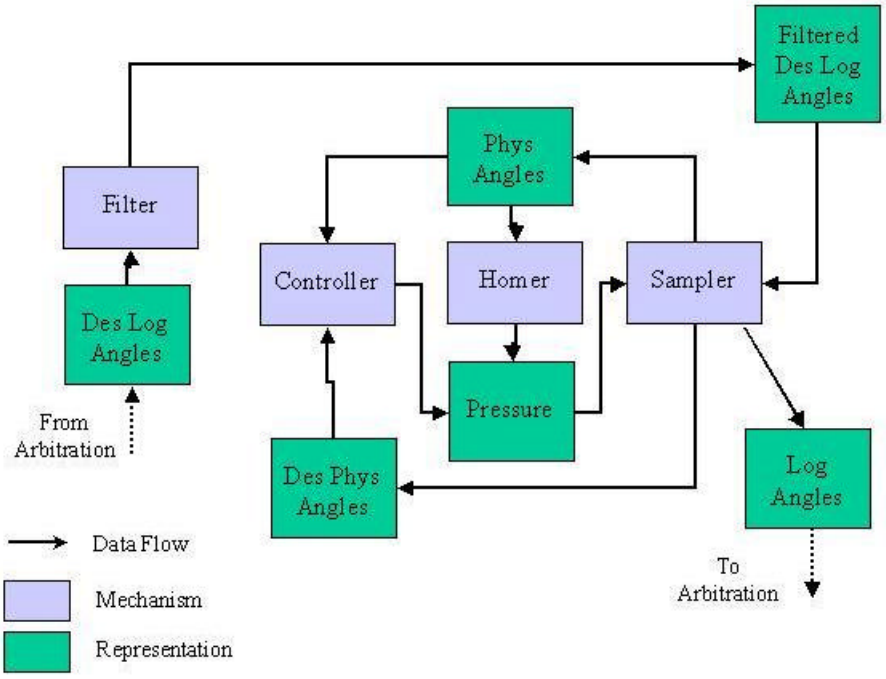

Figure 18. Soft Arm Agent Control Loop [Alford 1999, 2.2]

Controller

The most recent controller for ISAC was developed by S. Northrup in 2001. Northrup's motivation for creating a new controller was to design a humanoid robot whose arm movements were smooth and safe, particularly for fast goal-directed motion.

These improvements would enhance the overall experience for humans when interacting with ISAC [pp. 7-9].

Northrup's controller was biologically inspired. Before the control paradigm is presented, a brief description of human arm EMG activation levels is given [pp. 72]. Based on Yamazaki's work [Yamazaki 1995] it was shown that for quick arm motions, there is reciprocal activation in antagonistic muscles that is usually followed by co-activation of the muscles when the movement terminates. Reciprocal activation is also known as triphasic activation and it describes the agonist-antagonist-agonist sequence of EMG signals in the muscle. It is important to decompose motion into horizontal and vertical planes. EMG activation levels differ since different forces are at work. For instance, in the horizontal plane, once motion ceases the EMG signal disappears; but according to Flanders [1996] in the vertical plane, a triphasic pattern is superimposed with tonic activation (activation levels are needed for quasi-static postural control) patterns. Flanders also outlined a method to decompose the signal into its tonic and phasic components, allowing the phasic portion of the signal to be analyzed. Since ISACs arm structure is based on antagonistic artificial muscles and the goal is to articulate reach motions in both the horizontal and the sagital plane, ISACs arms can be activated and controlled in a manner similar to those in humans.

Hence, ISACs arms are modeled after the tonic-plus-phasic control paradigm. EMG signals in humans are similar to those of pressure in McKibben actuators. As force increases in muscles, EMG activation increases, and the length of muscle decreases. In McKibben actuators, when force increases, pressure increases likewise, whilst the length of the rubber decreases. Hence, a mapping of the reaching motions and the pressure

levels of the rubbertuators is needed. The mapping and model was developed through experimentation by (a) evaluating the tonic activation levels for postural control, and (b) evaluating the phasic activation level for reaching movements in the vertical plane was superimposed [Northrup 2001, pp. 73-76]. Thus an open loop architecture that models the tonic and phasic signals as time sequences for the arm to perform reaching motions was implemented and is shown in figure 19.

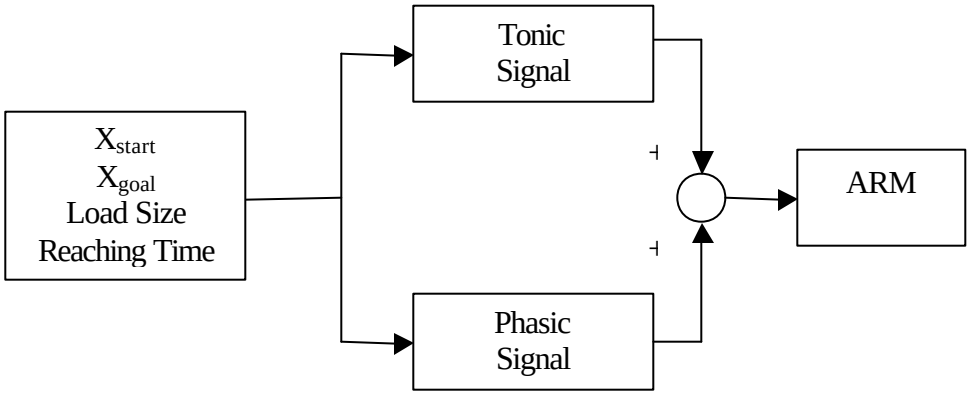

Figure 19. Open loop control architecture for controlling the reaching movement's of ISACs arms [Northrup 2001, pp. 76].

Reaching motions may employ a variety of sensory modalities for motor control – amongst those, visual and proprioceptive feedbacks are present in ISAC. However, if the desired motion is a fast-directed movement toward a target, there is not enough time for visual or proprioceptive feedback loops to be effective [pp. 74]. Thus, a feedforward control technique was chosen to overcome this problem. Northrup stated, "the research conducted for (his) this dissertation is a novel approach to the problem of reaching in a vertical plane with a nonlinear actuated robot arm" [pp.75]. To successfully implement the feedforward tonic-plus-phasic control paradigm, proprioceptive feedback needed to be incorporated [p. 84]. Based on the biological motor control hierarchy presented by

Crawford [1998] and shown below (figure 20) for simplicity it was shown that the long-loop error feedback lasts around 100 milliseconds.

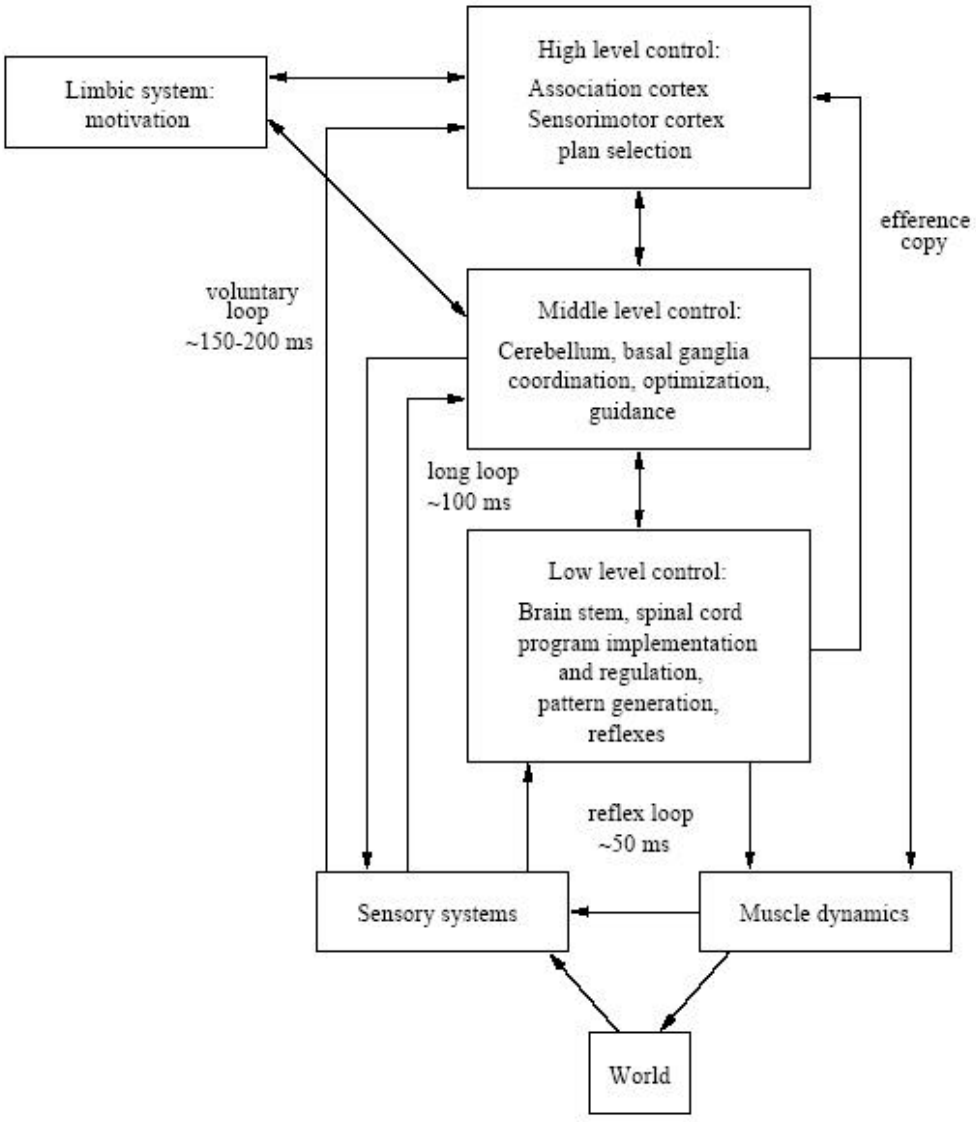

Figure 20. Motor Control Hierarchy [Crawford 1998, pp. 7].

Northrup referenced the error efferent signals after the feedback could occur. The controller compared the actual motion with the programmed one and if the difference exceeded an empirically calculated threshold, the feedback error controller would adjust

the motion. A block diagram of the final controller is shown below in figure 21. In it X_{start}, X_{goal}, are the initial and final goal positions, Reaching Time is the duration of the movement, and Load Size is the weight of the grasped object. For a given set of parameters, a motor program is known and is sent to the artificial muscles as a time sequence of the sum of the phasic and tonic activation levels. Also, after one hundred milliseconds the feedback loop was activated, see figure 21.

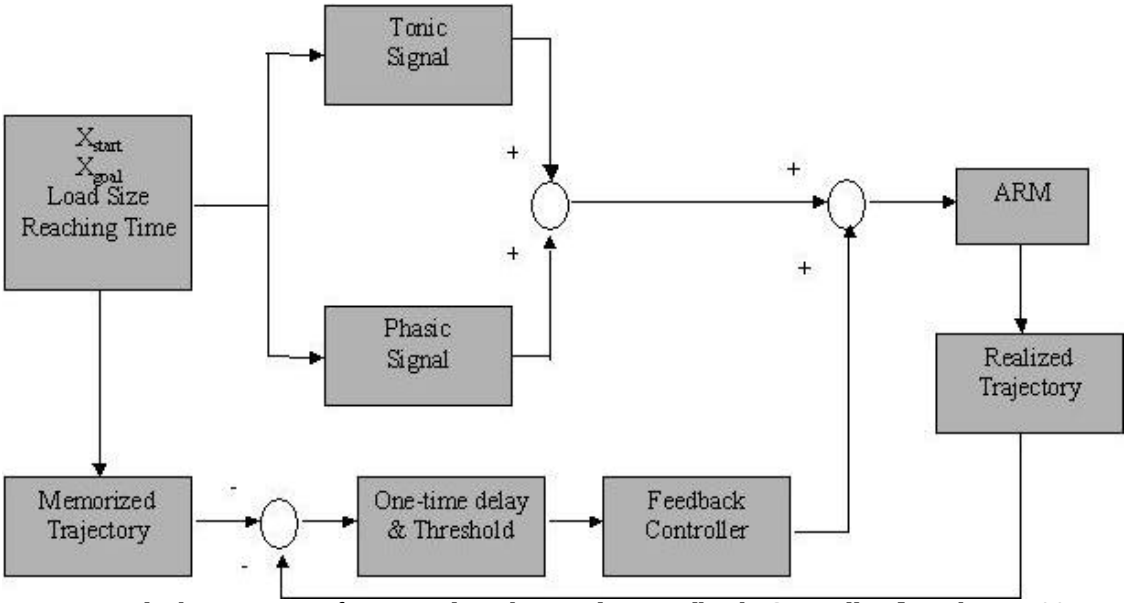

Figure 21. Block Diagram of Tonic-plus-Phasic Plus Feedback Controller [Northrup 2001, pp. 85].

Arbitration Loop

The arm as mentioned earlier can be articulated by providing goal positions in joint space or Cartesian space form. Cartesian commands are converted to the desired Cartesian position by means of an IMA object component called "CartArb." This location is converted to angles by inverse kinematics. The values are sent to the "KinLink"

component. This in turn is used to compute the desired logical angles, which are used by the control loop, see figure 22.

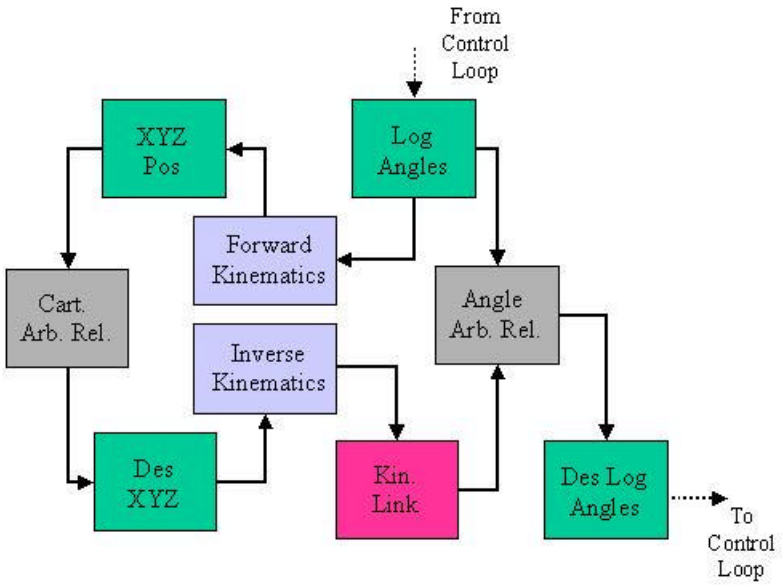

Figure 22. Soft Arm Arbitration Loop [Alford 1999, 2.3].

Similarly, the joint space angles can be converted to a Cartesian position by means of forward kinematics. Both Cartesian and logical angles can be sent to other agents through IMA relationships.

Hand

ISAC's hand has undergone changes to increase its control. The hand is a hybrid design that consists of a motored power thumb and forefinger and pneumatic distal fingers. The hand was implemented J. Christopher, and this description borrows from his work [Christopher 1998] and is shown in figure 23.

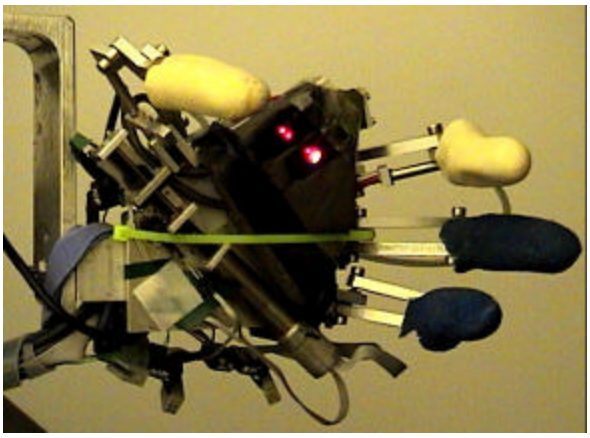

Figure 23. ISACs Hand.

The motored thumb and forefinger are driven by mechanical nuts. Also, the rest of the fingers use pneumatic actuators for additional control and increased number of degrees of freedom. A PC controller card specifies the desired pressure on gas valves that either open or close the hand.

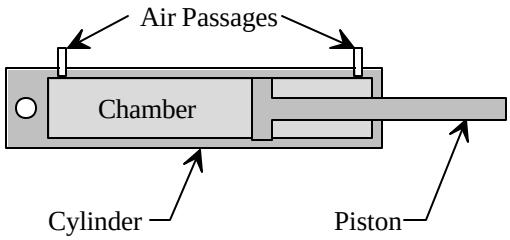

Figure 24. Pneumatic Cylinder [Alford 1999, 4.2].

Additionally, the hand possess proximity sensor to gain timing and position data. These sensors allow for more informed grasping decisions. The photoelectric sensors are placed in the palm of the robot. This allows the robot to sense the workspace environment for all grasping situations. Two sensors are used in the palm; one measures distances of 100mm and the other measures distances of 10mm. See figure below.

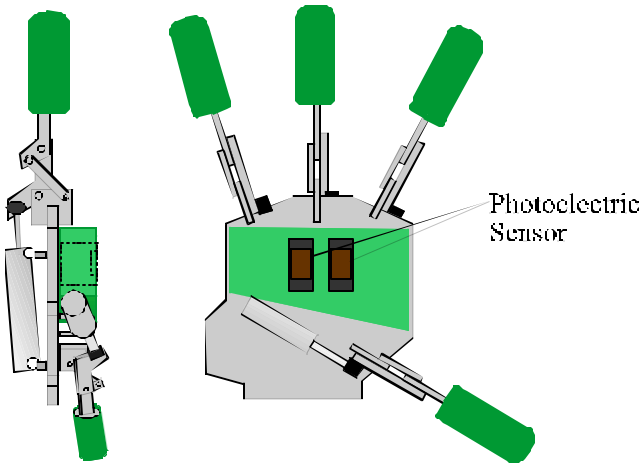

Figure 25. Photoelectric Sensors [Alford 1999, 4.5].

CHAPTER V

RESEARCH: SENSORY-MOTOR CONTROL

The research focus of this thesis is based on sensory motor coordination. The theory behind sensory motor coordination, from now on referred to as SMC, is reported.

As described by Cambron and Peters [2000], at its foundation, sensory motor coordination is the basis for behaviors for all animals. It behaves as a feeback loop through which an animal learns the environment producing a change in both. Motion in animals is produced by a contraction of the muscles, which is triggered by electrochemical signals that originate in the motor neuron circuits. Motion causes the environment to move with respect to the animal which results in changes in its sensory input. The sensory organs transduce energy from the environment into electrochemical signals that are carried by the circuitry of sensory neurons to produce a spatio-temporal representation of the world. There is an intimate relationship between the sensory signals and the motor signals, giving rise to the sense-act paradigm.

Similarly, SMC is useful for robots to experience and modify their environment. In machines, each sensor is modelled as an independent agent or module and yields basic motor reactions typically called “reflex reactions.” These reactions can be sequential or concurrent, each behavior can be activated or inhibited on the basis of the task and the context of the environment. Complex behaviors in turn are produced by combining several meta-level behaviors.

There is evidence to suggest that SMC serves as a foundation for higher level learning. The basic information obtained from the environment can be used to deliberate based on some goal a sequence of tasks to execute. As demonstrated by Peters et al. [2004], robots can learn from their own experience by constructing models of the dynamics of its own SMC data. Teleoperation is the means to data acquisition. In 1997, Pfeifer reported that both sensory and concurrent motor data could be represented by a vector-time series formed by clusters in a sensory motor state-space. Where, the locus of each cluster in the state-space would be equivalent to a meta-level behavior, giving rise to categorization of the environment for specific SMC events. When a robot performs a task and its SMC vector time-series is recorded, a smooth space-state trajectory is seen, partitioned by a series of jumps or clusters of information that demarcate different meta-level behaviors.

Peters et al. [2004], then showed that by learning a set ot trajectories that cover the extreme points of the workspace, the task can then be executed autonomously by the robot under many conditions. Even in new situations, the superposition of basic behaviors is used to give rise to the new task.

It is with this research interest, that the task of putting together a functioning humanoid robot was done. By using ISAC to emulate human reactions to sensory information, two results may be achieved: to learn more about the environment and react to it in a human-like fashion, and to through experience yield autonomous behaviors. A description of the demonstrations implemented for ISAC now follows.

CHAPTER VI

AGENTS

As described earlier, the IMA architecture allows for the implementation of multiple agents. Each agent is composed of multiple components that provide the agents with specific functionality. A total of six agents were used: the Sound Agent, the Camera Agent, the Head Agent, the Hand Agent, the Arm Agent, and the Trajectory Agent. The function of each agent and its main components will now be presented.

Sound Agent

The Sound Agent is used to find the direction of the sonic cues provided when an agent in the environment emits any kind of noise. There is a predetermined threshold value, used to avoid any noise values from being detected. The agent's main functionality within the scope of this work is to present the direction of the sound cue. The cue is presented in the form of a pan angle, with the purpose that the camera head can respond to this representation in a natural way. The angles are provided in intervals of 15 degrees starting from the center. Angles in the clockwise direction are negative, while those in the counter-clockwise direction are positive, see table 3.

The angles are presented in an array-like form such that the pan angle is the third element of an array-like structure. An image of this is shown in figure 26.

Table 3. Sound Localization Angles.

Direction	Angle
Clockwise	-90
Clockwise	…
Clockwise	-15
Center	0
Counter-clockwise	15
Counter-clockwise	…
Counter-clockwise	90

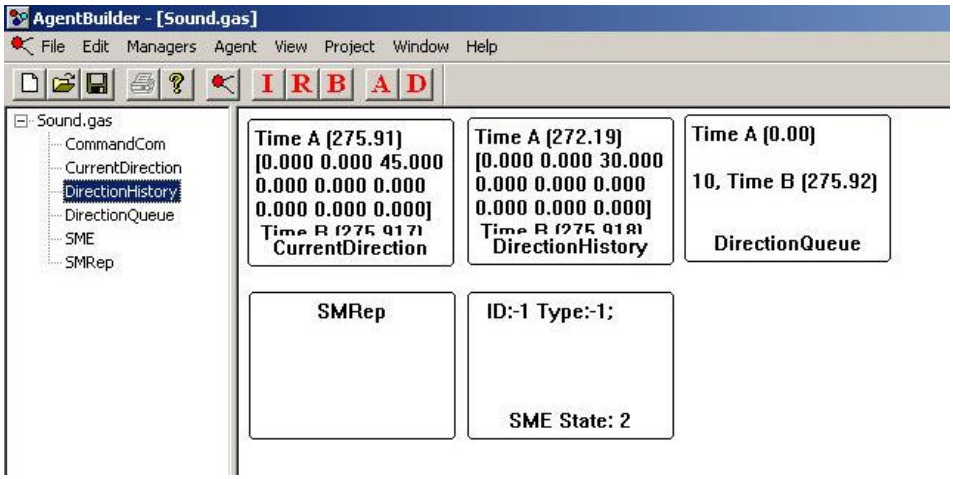

Figure 26. Sound Agent.

Appended in table 4, a description of each of the components in the agent is provided.

Table 4. Sound Agent Components

Component Name	Component Type	Purpose
CommandCom	Command Communicator	Sends events to other agents.
Current Direction	Vector Signal	Reports current sound cue direction
Direction History	Vector Signal	Reports past sound cue directions.
Direction Queue	Text Queue	Reports angle and time in text (for SES).
SME	State Machine Engine	Finite State Machine
SMRep	State Machine Representation	Interface to State Machine.

Camera Agent

The Camera Agent is used independently for both the right and the left cameras. The first function of the agent is to catch the images coming in from the cameras. This is shown in figure 27; depending on the desired function the color segmentation or motion detection techniques can be selected. Currently only one of these can be utilized at any one time. If segmentation takes place, the center of mass of the blob is calculated and returned as the relative position in the window.

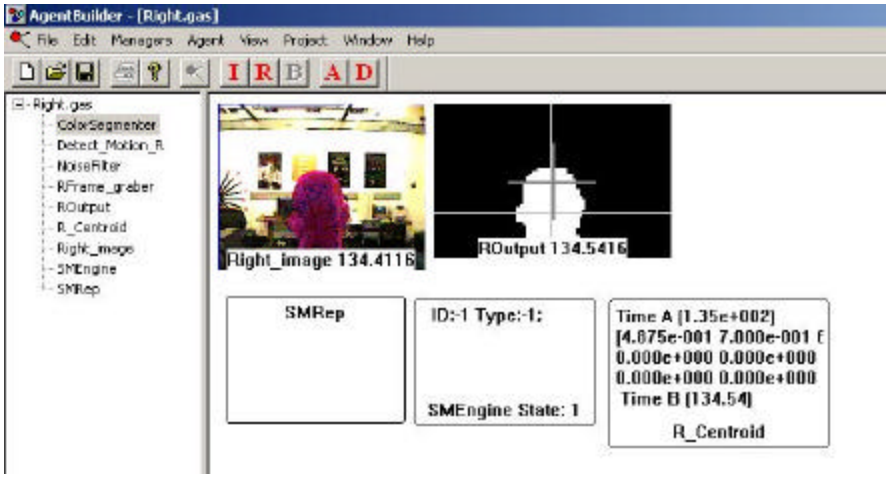

Figure 27. Camera Agent.

Appended in table 5, a description of each of the components in the agent is provided.

Table 5. Camera Agent Components.

Component Name	Component Type	Purpose
Color Segmenter	Mechanism	Segments according to color model.
Detect Motion	Mechanism	Segments according to motion.
Frame_grabber	Mechanism	Catches image from camera and sends it to a window in the GUI.
Output	Mechanism	Repository for processed image.
Centroid	Vector Signal	Displays location of segmented object.
Image	Mechanism	Repository for caught images.
Noise Filter	Intel IPL	Filters input image.
SME	State Machine Engine	Finite State Machine
SMRep	State Machine Representation	Interface to State Machine.

Head Agent

The head agent serves as the "brain" of these sensory-motor demonstrations. First, it will be described in the context of its relationship to the sound and Camera Agents, and then in the context of its relationship with the motor agents. The head agent receives the focus of attention from both of the Camera Agents. This is done through a simple proxy connection. The focus of attention in the cameras is then used in an attention network that provides a single fixation point in the form of camera coordinates. A side note is of consideration. ISAC underwent a recent pan-tilt unit change. The older unit worked under a slightly different set of coordinates, that is: [Left Verge, Right Verge, Pan, and Tilt].

The head agent has been revamped to handle the older set or the current set for the two independent pan-tilt units: [Left Pan, Left Tilt, Right Pan, and Right Tilt]. At any time an incoming set of data in the older form appears, the data is converted to the new form. Once the fixation point is acquired, there are several ways to move the head, either through direct mapping, or through saccadic or smooth pursuit movements. The choice of movement is determined by the user in advance, and is then established in the finite state machine of the agent. Additionally, by using the angle information obtained from sampling the pan-tilt units, the depth of the agent in the environment can be calculated in Cartesian coordinates. The latter serves as the transition point to the motor segment of the discussion. The Cartesian coordinate of the gazed target is the goal for all the motor activities that take place. The depth estimation information is passed to the Trajectory Agent. Essentially, the latter uses the coordinate to create its path in space, which is articulated by the Right Arm Agent. An image of the Head Agent is seen below.

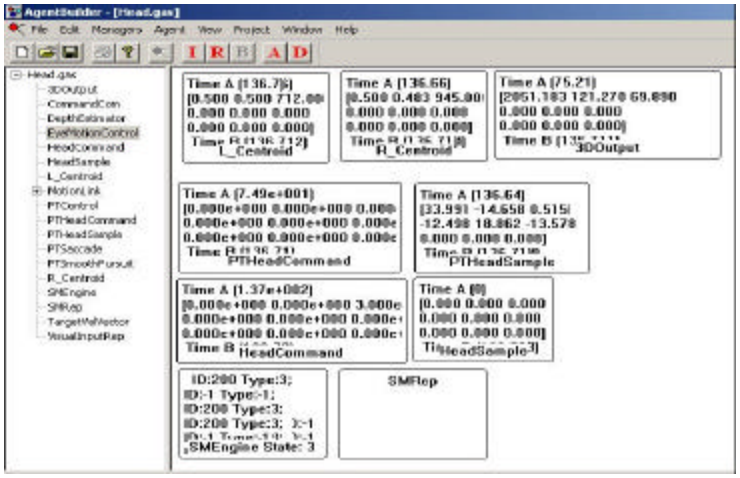

Figure 28. Head Agent.

Appended in table 6, a description of each of the components in the agent is provided.

Table 6. Head Agent Components.

Component Name	Component Type	Purpose
3DOutput	Vector Signal	Displays Cartesian location.
CommandCom	Command Communicator	Sends events to other agents.
Depth Estimator	Mechanism	Calculates depth of object.
EyeMotionControl	Mechanism	Calculates fixating point from incoming eye movements inputs.
Head Command	Mechanism	For former head format. Tells the pan-tilt unit to move.
Head Sample	Mechanism	For former head format. Sample the pan-tilt unit.
L-Centroid	Vector Signal	Incoming position of object from left camera
Motion Link	Motion Link	Links agent to the arm agent.
PTHeadCommand	Mechanism	For current head format. Tells the pan-tilt unit to move.
PTHeadSample	Mechanism	For current head format. Sample the pan-tilt unit.
PTSaccade	Mechanism	Performs saccade calculations.
PTSmoothPursuit	Mechanism	Performs Smooth Pursuit calculations.
R_Centroid	Vector Signal	Displays positions from right camera.
SME	State Machine Engine	Finite State Machine
SMRep	State Machine Representation	Interface to State Machine.
TargetVelVectory	Dynamic Vector	Calculates velocity of target.
VisualInputRep	Mechanism	Performs neural net calculation.

Hand Agent

The Hand Agent in line with its hardware properties has two different grasps mechanisms, a fast grab and a slower but more precise grab. The former does not use the motored fingers, whereas the latter does. The choice of grasp is selectable and can be registered in the finite state machine. Additionally, the hand contains proximity sensors that indicate the presence of a nearby object or obstacle. This information is used to prevent false (empty) grasps. Typically, the hand is called to use at the completion of an arm trajectory. The agent interface is shown below.

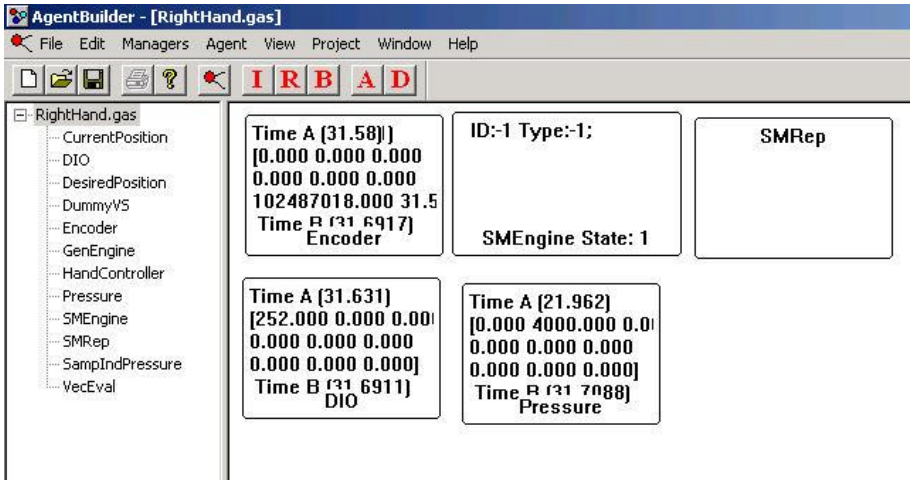

Figure 29. Right Hand Agent.

Appended in table 7, a description of each of the components in the agent is provided.

Right Arm Agent

The Arm Agent contains components that control the arm and perform the kinematic manipulations to articulate the appropriate motion in the arm.

Table 7. Hand Agent Components.

Component Name	Component Type	Purpose
Current Position	Vector Signal	Displays actual logical angles.
DIO	Mechanism	Reports the digital input/output signal.
Desired Position	Mechanism	Displays logical angles.
Encoder	Vector Signal	Displays encoder values.
GenEngine	General Engine	Runs the sampler mechanism.
SME	State Machine Engine	Finite State Machine
SMRep	State Machine Representation	Interface to State Machine.
SampIndPressure	Mechanism	Controller for hand.
VecEval	Mechanism	Measures for proximity of objects in hand.

Goal locations can be passed to the arm in two forms: Cartesian coordinates and joint angles. In the work presented here, all goal locations were specified in Cartesian coordinate form. See figure 30.

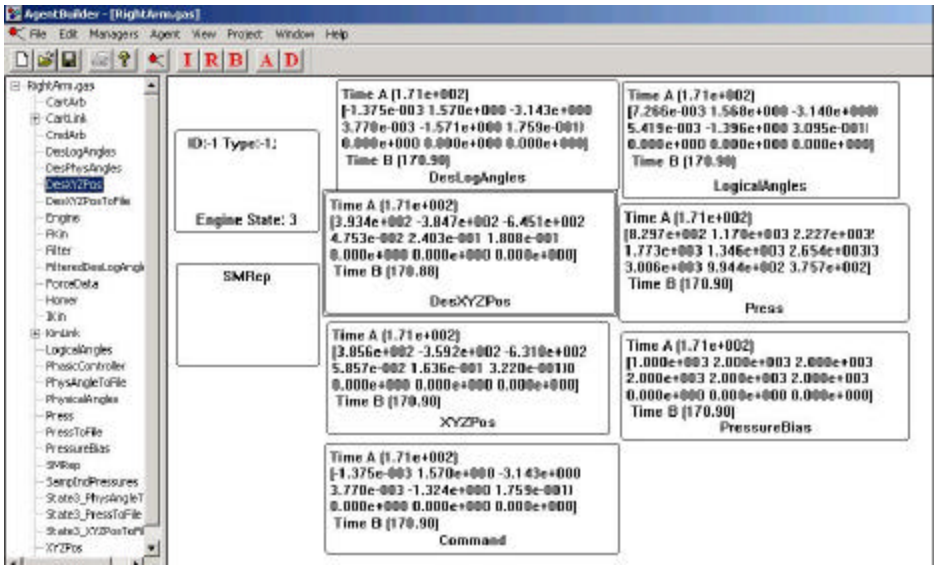

Figure 30. Right Arm Agent.

Appended in table 8, a description of each of the components in the agent is provided.

Table 8. Right Arm Agent Components[Northrup 2001, pp. 161]

Component Name	Component Type	Purpose
CartArb	Arbitration	Allow Cart command to arm from other agents.
CartLink	Motion Link	Cart command to input, activation, & feedback.
CmdArb	Arbitration	Allow Logical Angle command. To arm from agents.
DesLosAngles	Vector Signal	Transmit desired logical angle data.
DesPhysAngles	Vector Signal	Transmit desired physical angle data.
DesXYZPos	Vector Signal	Transmit desired Cartesian position data.
DesXYZPosToFile	Mechanism	Record desired Cartesian position to data.
Engine	General Engine	Allows automatic mechanism activation.
Fkin	Mechanism	Compute forward kinematics.
Filter	Mechanism	Infinite impulse response filter for des los angles.
FilteredDesLosAngles	Vector Signal	Transmit filtered des log angle data.
ForceData	Vector Signal	Transmit force-torque sensor data.
Homer	Mechanism	Homing mechanism.
Ikin	Mechanism	Computes inverse kinematics.
KinLink	Mechanism	Transmit data as part of arbitration.
LogicalAngles	Vector Signal	Transmit logical angle data.
Phasic Controller	Mechanism	Controller for ISACs arms.
PhysAngleToFile	Mechanism	Record physical angle data.
PhysAngles	Vector Signal	Transmit physical angle data.
Press	Vector Signal	Transmit pressure delta. (Either 6 delta-P or 12 individual pressures.

PressToFile	Mechanism	Record pressure data.
Pressure Bias	Vector Signal	Transmit pressure data.
SMRep	Mechanism	State machine mechanism.
SampIndPressures	Interface to State Machine.	Soft Arm Sampler. Input/Output of physical data from and to ISAC's arm.
State3_PhysAngleToFile	Interface to State Machine.	Record physical angle data during state 3.
State3_PressToFile	Interface to State Machine.	Record pressure data during state 3.
Sate3_XYZToFile	Interface to State Machine.	Record Cartesian position data during state 3.
XYZPos	Vector Signal	Transmit Cartesian position data.
XYZPosToFile	Mechanism	Record Cartesian position data.

Trajectory Agent

The trajectory agent is used to generate and play the exact trajectories for the arm. The arm motions are implemented based on a starting and ending points (intermediate points for specific routing can be used as well); the requested duration of trajectory; and the parameter type (either joint angles or Cartesian coordinates). For the experiments described here, the starting point always was the home position of the arm, and the final point was obtained from the depth estimation component of the Head Agent. Hence a Cartesian based trajectory was created from the home position to the location of the desired object. The duration of the motion was established at two seconds. The number of trajectories varied depending on the task and goal. An iterative motion where the arm repeatedly reaches to the desired object could easily be generated. The agent is shown in figure 31.

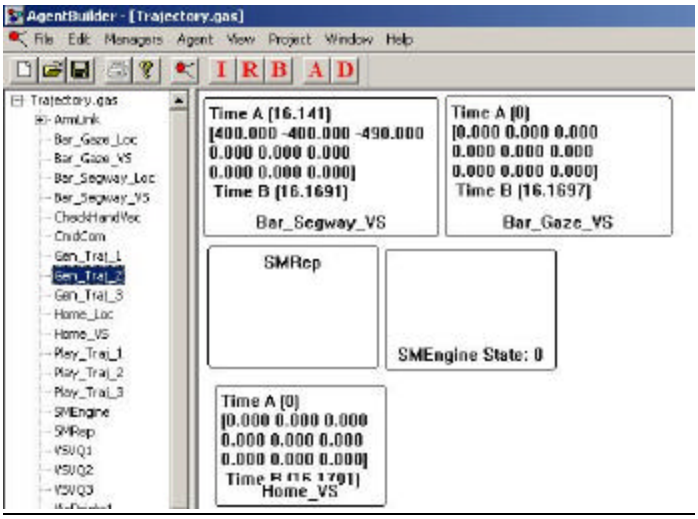

Figure 31. Trajectory Agent.

Appended in table 9, a description of each of the components in the agent is provided.

By implementing these six autonomous agents and meaningfully interconnecting them a wide variety of sensory-motor behaviors can be achieved. An effort to produce a wide range of behaviors with different representations of cognition is presented in the next chapter.

Table 9. Trajectory Agent Components.

Component Name	Component Type	Purpose
ArmLink	MotionLink	Links agent with arm.
Bar_Gaze_Loc	Mechanism	Starting point for first trajectory.
Bar_Gaze_VS	Vector Signal	Point displayed.
Bar_Segway_Loc	Mechanism	Starting point for second trajectory.
Bar_Segway_VS	Vector Signal	Point displayed.
CheckHandVec	Mechanism	Proximity sensor.
CmdCom	Command Communicator	Sends events to other agents.
Gen_Traj	Mechanism	Generates trajectories given starting- and end- points.
Home_Loc	Mechanism	End point for last trajectory.
Home_VS	Vector Signal	Displays final position
Play_Traj	Mechanism	Moves arm wrt trajectory.
SME	State Machine Engine	Finite State Machine
SMRep	State Machine Representation	Interface to State Machine.
VSVQ	Mechanism	Transformation from vector signal to vector queue.
ViaPoints	Vector Queue	Intermediate points in a trajectory.

CHAPTER VII

DEMONSTRATIONS

Five different demonstrations were implemented with the goal of conveying cognitive behaviors in the humanoid robot at the Cognitive Robotics Lab at Vanderbilt University. The demonstrations begin with a low degree of difficulty and increase with each consecutive routine. As stated before, the objective is for the robot to learn about its world and to interact with it in a meaningful way.

Goals

The goal of the demonstrations is to produce basic behaviors by achieving sensory fusion and articulating basic motion in the robot. The latter is better defined in terms of human reactions. Humans possess a number of natural responses to stimuli in their environment – all of which can be affected by the attention factor in the human. When a sudden noise is heard by the ear, the human tends to look at the area of activity and tries to find the source of the sound. Sometimes this behavior can be accompanied by a desire to reach-and-grasp action to further analyze the source.

Implementation

With this in mind, five different demonstrations were implemented to show each of the building blocks for these instinctive behaviors. These are now presented in the following list:

Table 10. Description of demonstrations.

Demonstration	**Description**
Audio + Head Motion	Audio cues followed by camera-head motion.
Audio + Head Motion (Color)	One audio cue followed by camera-head motion and gazing through color segmentation.
Audio + Head Motion (Motion)	An audio cue followed by camera-head motion and gazing through motion detection.
Audio + Head Motion + Arm	An audio cue followed by camera-head motion and gazing through color segmentation. This then triggers a single reach motion.
Audio + Head Motion + Arm	An audio cue followed by camera-head motion and gazing through color segmentation. This then triggers a smooth reach-and-grasp motion.

Seven different IMA agents were used to produce the behaviors. Each demonstration required a different number of agents, ranging from two for the simplest performance, to seven for the most complex. Each agent has specific functionality and normally shares information with other agents.

The first demonstration displays a relationship between sonic cues and attention. Two agents were active in this demonstration: the Head Agent and the Sound Agent. The Sound Agent was designed to output the angle of a given sonic cue at intervals of 15 degrees. The data was passed to the Head Agent through an array-based representation, where the pan angle was in the third element of the array. The scenario was set so that at any point in time a given audio cue was received, the pan-tilt unit would pan to the appropriate direction. This behavior was set to occur continuously.

The second demonstration was based on the same principle. However, the aim was to involve vision and achieve a gazing behavior from the cameras. Four agents participated for this second display: the Sound Agent, the Head Agent, and two Camera Agents. Color segmentation was used as the image processing technique to retrieve

information from the environment. Specifically, the color segmentation component was loaded with a model to find skin tones; hence, the camera-head fixated on the face of the person in front of it. An important step to note here is the transition for the pan-tilt unit in following commands from the Sound Agent to commands from the Camera Agents. A simplistic time-based event could have been used to change the mode of the pan-tilt unit after some time, yet this principle would not have been consistent with the desire to emulate human conduct. Humans turn and seek simultaneously. Hence, in an effort to assimilate the human system closer, an agent dispatch command was used. Whenever an audio cue was sensed successfully and the pan-tilt unit turned towards it, an event would be triggered causing the Head Agent state machine to shift triggering two additional events: 1) the Camera Agents are sent a command to initiate color segmentation, and 2) the pan-tilt unit begins tracking of the point of attention. The camera-head can fixate on any element of the environment that fits to the color model assigned. As mentioned before, this color model is trained with specific data to empower the cameras to detect a desired color. At this point, the camera-head will track the desired target in an uninterrupted fashion.

The third demonstration is similar to the one above. The difference is found on the gazing technique. Instead of using color segmentation to find a target in the environment, a motion detection technique is used. The latter will return the image position of the center of mass of a moving blob. Ideally, a robotic vision system would emulate that of the human being and simultaneously perform a number of image processing techniques. However, that is one limitation in our system, and thus two separate exhibitions were done to show these differences.

The fourth run includes a more attractive behavior: reaching. Thus, an extra agent is needed for the control of the arm. The building blocks for this performance to take place are the same as the ones in the second demonstration. It is now fitting to mention the capability of the Head Agent to calculate the Cartesian coordinates. When the pan-tilt unit fixates on an object, the Depth Estimator component outputs an x, y, and z location. This Cartesian coordinate data is directly linked to the Arm Agent; hence, the arm moves at the same time the coordinates are calculated. The motion of the arm is characterized by a single rapid motion to the indicated point.

Finally, the fifth demonstration was implemented. This one is particularly characterized by a smooth trajectory for the arm and a grasp behavior. Two additional agents are used for this implementation – that is the Trajectory Agent and the Hand Agent. The capacity of the Trajectory Agent allows the arm to undergo a complex motion within its workspace. An endless number of movements can be produced through this agent. These motions are linear and are created based on starting and ending points. The points can be arbitrary or can be obtained through either the Depth Estimator component mentioned above, or the location of the arm through the XYZPosition component, which fulfills the same purpose as the Depth Estimator object – a Barney doll was used as a goal. In this demonstration, a complex motion consisting of four basic motions was utilized. The first motion began at the home position of the arm and then moved a few centimeters forward. The second motion continued from that current location to the coordinates given by the Depth Estimator component. At this point, the Barney doll is within reach of the hand. At this point, the state machine commands the hand to check through its proximity sensors the presence of an object. If true, a command is sent to the

Hand Agent to tell it to close its grip. After completion, the third motion resumes, taking Barney to the first starting point and then releasing him there. Finally, the arm is commanded to finish its motion at its home position.

Limitations

The inherent limitations of the demonstrations lie in the fact that they are reactive demonstrations. Any behaviors exemplified by ISAC are fixed responses to sensory input. Since the system lacks higher-level commands or dynamic attention points, ISAC can only perform single sequential tasks – turn to a given noise, fixate on a predetermined color object, and perform a reach-and-grasp motion on the seen object.
The system at this cannot convey an adaptive behavior based on higher-level commands. It is not able to distinguish between desirable sounds and noise, it cannot autonomously gaze at different objects based on a change in the scenario, nor can it decide if it should hinder a reach-and-grasp motion if it were not appropriate.

Finally, a complete integration and cooperation of auditory and visual sensory information is not in place. The humanoid cannot simultaneously use information from both inputs to paint a picture of the environment; rather, a sequential input string takes place, where auditory inputs trigger visual inputs, which in turn trigger articulate motion in the arm.

Enhancing the system through further development and integration with higher levels of abstraction will be discussed in the FUTURE WORK section of this thesis.

CHPATER VIII

CONCLUSIONS

The humanoid robot successfully utilized entry data from the environment to articulate motion in a reflexive manner. ISAC reliably detected the presence of agents in the world and displayed sensory-coupled reactions that assimilate those of a human baby. In the presence of noise in the environment, the robot was able to pan towards them in search of objects of interest. This reflex action sets the stage for a meaningful interaction between the machine and the human. When humans address the robot, the latter immediately turns toward the direction of speech empowering the robot to learn more about the human through voice commands and later on through visual cues. Similarly, the robot through experience can strengthen the associations between the given auditory information and the panning motion. Additionally, when the robot was directed to recognize objects whether through color or motion features, the humanoid effectively panned, fixated, and tracked its goal objects. Hence, once vision is at work with a point of attention potential salient features of the object of interest become available – color, motion, texture, shape, size, and emotion. In doing so, the robot has been empowered to analyze these features and gain understanding of its world. By using information from audio and vision, associations between specific sounds and visual features can start to occur. Finally, vision allows reach-grasp behaviors to occur. In reaching and grasping, the robot gains the ability to fetch and bring objects to a fixation point for further analysis. This type of interaction necessarily influences the environment, and dynamic

and continuous associations between motor, visual, and auditory cues can be learned and provide a deeper understanding of associations in the environment to the robot. The ground work has now been set to allow ISAC to develop more complex interactions with its world, whether it is by coupling with higher level agents, learning through its own sensory-motor vector space representations, or reacting to sensory information.

Future Work

The breadth of research for this field of study is vast and unknown in many of its branches. Future research takes many directions and all of them need to be explored. The proficiency of all software systems can be vastly enhanced in an attempt to resemble those of the human system. At the base of all these, is the current software architecture used. Although scalable, encapsulated, and flexible, it still falls short in allowing for multiple fast parallel systems to act and interact together. Regarding the sensory part, the auditory system in the humanoid ought to be capable of detecting sound at a fine-grained level. Along with sound detection, its natural companion would be speech recognition and speech emission. In emulating humans, language comprehension and communication is essential for interaction. Of course, the visual system, one of the hardest to develop, calls out for much improvement in attempting to be more human in nature. The arm boasts a capable controller yet it lacks the adaptive nature that a human has. If the trajectory is to be changed at mid-flight, the current controller would not be able to change on the fly. An adaptive controller is necessary to modulate the actuator motion. Finally, the current hand (which was made in house) is unreliable and provides little functionality. It breaks down often and performs abrupt grasps with no measure of

goodness of grab. A dexterous and sensitive hand is desired, UK's Shadow Robot Company [Shadow 2004] possesses a very dexterous hand. Such hand would enhance the robot's world experience by being able to easily grasp-and-release, push-and-pull, and have tactile sensations.

Another branch for future work concerns the robot's own sensory-motor experience. Extensive research is currently being performed at Vanderbilt University's Center for Intelligent System. The work of Campbell [2003], Ao, [2003], and Peters et al. [2004] represent efforts to more clearly understand the nature of the vector-space representations of sensory-motor coupled data. By properly understanding the clustering of data in the vector space, superposition of clusters can lead to effective execution of complex behaviors that would interact with the world in a natural way.

Finally, the additions of attention and higher level commands guided by decision-making systems would push the robot to a cognitive state where intelligence can be explored and developed. Efforts to reproduce cognition are also being pursued at the Center for Intelligent Systems [Kawamura et al. 2004] and others such as Fitzpatrick [2003], Scassellati [2000] and this area of study promises to be a breakthrough in the field of robotics. The study of humanoid robotics and all of its challenges is truly an interesting field to study. Technological advancement will come in the future and exciting breakthroughs will drive the effort to develop cognitive robots closer to reality.

APPENDIX A

STATE MACHINE'S

Sound Agent State Machine

```
State 0: (No Comment)
Timeout: 10000, Period:100

Activities

Transitions
  Next State: 2, activate
    Event: 2, 2
  Next State: 1, activate
    Event: 1, 1


State 1: Report Sound Cue in Sensory Egosphere form.
Timeout: 10000, Period:400

Activities
  Trigger 0: CommandCom(2,0), listen

Transitions
  Next State: 2, deactivate
    Event: 2, 2
  Next State: 0, deactivate
    Event: 1, -1


State 2: Report Sound Cue in LV RV P T form.
Timeout: 1000, Period:100

Activities
  Trigger 0: CommandCom(2,1), Retrieve angle in LRPT form

Transitions
  Next State: 0, deactivate
```
Event: 1, -1

Camera Agent State Machine

```
State 0: (Catch image)
Timeout: 500, Period:100
```

```
Activities
  Trigger 0: LFrame_graber(1,0), Catch image

Transitions
  Next State: 1, Do color segmentation
    Event: -1, -1


State 1: (Do color segmentation)
Timeout: 1000, Period:100

Activities
  Trigger 0: LFrame_graber(1,0), Catch image
  Trigger 1: ColorSegmenter(0,0), Do color segmentation
```

Head Agent

```
State 0: (Homing Position)
Timeout: 100, Period:100

Activities
  Trigger 0: EyeMotionControl(300,0), Send to home position

Transitions
  Next State: 1, Throw queue to start sound agent
    Event: 0, 1


State 1: (Throw queue to start sound agent)
Timeout: 200, Period:100

Activities
  Trigger 0: CommandCom(2,0), Throw queue to start sound agent

Transitions
  Next State: 2, Move towards sound queue
    Event: -1, -1


State 2: (Move towards sound queue)
Timeout: 2500, Period:100

Activities
  Trigger 0: PTControl(6,0), Move towards sound queue

Transitions
  Next State: 3, Throw event to start cameras and smooth pursuit
    Event: 2, 3


State 3: (Throw event to start cameras and smooth pursuit)
Timeout: 200, Period:100

Activities
```

```
  Trigger 6: CommandCom(2,3), Throw event 100, 100 to start arm
  Trigger 5: DepthEstimator(3,1), Display x,y,z,depth
  Trigger 3: EyeMotionControl(301,0), Smooth pursuit to target
  Trigger 4: EyeMotionControl(400,0), Display motor positions
  Trigger 2: EyeMotionControl(101,0), Sample motor positions
```

Hand Agent

```
State 0: ((null))
Timeout: 1000, Period:90

Activities

Transitions
  Next State: 1, (null)
    Event: 1, 1
  Next State: 3, (null)
    Event: 1, 3
  Next State: 2, (null)
    Event: 1, 2
  Next State: 4, (null)
    Event: 1, 4


State 1: ((null))
Timeout: 100, Period:90

Activities
  Trigger 1: HandInterface(1,0), (null)

Transitions
  Next State: 0, (null)
    Event: 1, -1
  Next State: 0, (null)
    Event: -1, -1


State 2: ((null))
Timeout: 5000, Period:100

Activities
  Trigger 0: VecEval(0,0), (null)

Transitions
  Next State: 0, (null)
    Event: 1, -1
  Next State: 3, (null)
    Event: 1, 3


State 3: ((null))
Timeout: 200, Period:100

Activities
```

```
  Trigger 0: HandInterface(3,0), (null)

Transitions
  Next State: 0, (null)
    Event: -1, -1


State 4: ((null))
Timeout: 10000, Period:100

Activities
  Trigger 0: PhSen(4,0), (null)

Transitions
  Next State: 0, (null)
    Event: 1, -1
  Next State: 5, (null)
    Event: 169, 5


State 5: ((null))
Timeout: 4000, Period:1000

Activities
  Trigger 0: HandInterface(3,0), (null)

Transitions
  Next State: 6, (null)
    Event: -1, -1


State 6: ((null))
Timeout: 4000, Period:1000

Activities
  Trigger 0: HandInterface(1,0), (null)

Transitions
  Next State: 4, (null)
    Event: -1, -1
```

Right Arm Agent

```
State 0: (Initialize and home)
Timeout: 10000, Period:100

Activities
  Trigger 3: SampIndPressures(0,0), Hardware I/O
  Trigger 2: Homer(0,0), Compute Homing Pressures
  Trigger 1: SampIndPressures(1,0), Disable Z-Masking
  Trigger 0: PhasicController(1,1), Read Data File

Transitions
  Next State: 1, Manual event
    Event: 1, 1
```

```
  Next State: 1, Hardcoded event from Homer
    Event: -2, -2
  Next State: 2, Manual event
    Event: 1, 2


State 1: (Go to initial angles)
Timeout: 10000, Period:60

Activities
  Trigger 2: SampIndPressures(0,0), Hardware I/O
  Trigger 3: PhasicController(0,0), Control Law
  Trigger 1: SampIndPressures(5,1), Output "home" angles
  Trigger 0: SampIndPressures(2,0), Enable Z-Masking

Transitions
  Next State: 2, Hardcoded event from sampler (when home reached)
    Event: 2, 2
  Next State: 2, Timeout
    Event: -1, -1


State 2: (Compute initial cartesian position)
Timeout: 5000, Period:100

Activities
  Trigger 0: FKin(2,0), Forward kinematics
  Trigger 2: IKin(3,0), Inverse kinematics
  Trigger 3: CmdArb(3,0), Inform current pos
  Trigger 1: CartArb(3,0), Inform client agents of current cartesian pos

Transitions
  Next State: 3, Timeout
    Event: -1, -1


State 3: (Non-Linear Controller, closed-loop)
Timeout: 10000, Period:75

Activities
  Trigger 2: CmdArb(0,1), Arbitrate Angle Commands
  Trigger 8: State3_XYZPosToFile(2,1), No Comment
  Trigger 11: PhasicController(1,2), Read Current Pressures
  Trigger 5: IKin(3,0), calc inverse kinematics
  Trigger 7: DesXYZPosToFile(2,1), No Comment
  Trigger 4: CartArb(0,1), CartArb
  Trigger 3: FKin(2,0), calc forward kinematics
  Trigger 1: SampIndPressures(6,0), Hardware I/O with 12 Pressures
  Trigger 10: State3_PhysAngleToFile(2,1), No Comment
  Trigger 6: Filter(0,0), IIR Filter on des log angles
  Trigger 9: State3_PressToFile(2,1), No Comment
  Trigger 0: PhasicController(0,1), Non-Linear Control Law (12
pressures)

Transitions
  Next State: 5, Manual Transition to state 5
    Event: 3, 5
```

```
  Next State: 1, (null)
    Event: 1, 1
  Next State: 4, Manual Transition to state 4
    Event: 3, 4


State 4: (Phasic Controller)
Timeout: 10000, Period:20

Activities
  Trigger 4: CmdArb(0,1), Arbitrate angle commands
  Trigger 10: XYZPosToFile(2,1), No Comment
  Trigger 1: PhasicController(1,2), Read Current Pressures
  Trigger 5: FKin(2,0), compute forward kinematics
  Trigger 6: CartArb(0,1), No Comment
  Trigger 3: SampIndPressures(6,0), use 12 individual pressures to
control arm
  Trigger 9: PhysAngleToFile(2,1), No Comment
  Trigger 2: PhasicController(1,0), Compute Pressures
  Trigger 7: IKin(3,0), compute inverse kinematics
  Trigger 11: PressToFile(2,1), No Comment
  Trigger 8: Filter(0,0), IIR on des logical angles

Transitions
  Next State: 5, Manual Transition
    Event: 4, 5
  Next State: 5, hardcoded event from PhasicController, end of movement
    Event: -100, -100
  Next State: 3, Manual Transition
    Event: 4, 3


State 5: (Phasic Controller Reset)
Timeout: 1000, Period:100

Activities
  Trigger 0: PhasicController(1,3), Reset counter, phasic move flag,
dispatch (-100,-300)

Transitions
  Next State: 3, Manual Transition
    Event: 5, 3
  Next State: 3, Hardcoded event from PhasicController
    Event: -100, -300
```

Trajectory Agent

```
State 0: (Initialization)
Timeout: 500, Period:100

Activities
  Trigger 0: Bar_Segway_Loc(2,0), Load "Barney's" position from file
  Trigger 1: CmdCom(2,1), Throw event so that the hand opens up

Transitions
```

```
  Next State: 10, Go to Ready
    Event: 100, 100


State 10: (Initialization)
Timeout: 200, Period:100

Activities
  Trigger 1: VSVQ1(1,0), Pass x,y,z location from VS to VQ for 1st
trajectory

Transitions
  Next State: 11, Generate the trajectory motion
    Event: -1, -1


State 11: (Generate the 1st trajectory after event is thrown by the
user!)
Timeout: 200, Period:100

Activities
  Trigger 1: ArmLink(0,3), Reset Cmd and Arb vectors
  Trigger 0: Gen_Traj_1(0,0), Generate the trajectory file based on the
starting and final points

Transitions
  Next State: 20, Throw event 100,100 to Initialize trajectory player
    Event: -1, -1


State 20: (Initialize Trajectory Player)
Timeout: 100, Period:100

Activities
  Trigger 0: Play_Traj_1(1,0), Initialize the trajectory player

Transitions
  Next State: 21, Play the trajectory file
    Event: -1, -1


State 21: (Play the 1st trajectory)
Timeout: 6000, Period:100

Activities
  Trigger 0: Play_Traj_1(0,0), Play the trajectory file and reach
towards Barney

Transitions
  Next State: 50, Load data for 2nd trajectory
    Event: -1, -1


State 30: (Wait for palm sensor to check for the presence of Barney)
Timeout: 100, Period:100

Activities
```

```
  Trigger 0: CheckHandVec(0,0), Check for the presence of Barney to
later close the hand

Transitions
  Next State: 40, Close hand
    Event: -1, -1


State 40: (Close hand)
Timeout: 500, Period:100

Activities
  Trigger 0: CmdCom(2,0), Throw event to the hand agent to close the
hand
  Trigger 1: CheckHandVec(0,0), Check until hand is closed
  Trigger 2: ArmLink(0,3), Reset Cmd and Arb vectors

Transitions
  Next State: 70, Load information to generate the 2nd trajectory motion
    Event: -1, -1


State 50: (Load the location for the gazing point)
Timeout: 500, Period:100

Activities
  Trigger 0: Bar_Gaze_Loc(2,0), Load the x,y,z location from the file to
gaze at Barney

Transitions
  Next State: 51, Pass the information from the VS to the VQ
    Event: -1, -1


State 51: (Load information to generate 2nd trajectory motion)
Timeout: 200, Period:100

Activities
  Trigger 1: VSVQ2(1,0), Pass x,y,z location from VS to VQ for 2nd
trajectory

Transitions
  Next State: 60, Generate the 2nd trajectory motion
    Event: -1, -1


State 60: (Generate 2nd trajectory motion)
Timeout: 200, Period:100

Activities
  Trigger 1: ArmLink(0,3), Reset the Cmd and Arb vectors
  Trigger 0: Gen_Traj_2(0,0), Generate the 2nd trajectory motion

Transitions
  Next State: 61, Throw event 200,200 to move arm up for ISAC to gaze
    Event: -1, -1
```

```
State 61: (Initialize the 2nd trajectory motion)
Timeout: 100, Period:100

Activities
  Trigger 0: Play_Traj_2(1,0), Initialize the trajectory motion to move
next

Transitions
  Next State: 62, Play the file
    Event: -1, -1


State 62: (Move arm up for ISAC to gaze)
Timeout: 4000, Period:100

Activities
  Trigger 0: Play_Traj_2(0,0), Move arm up for ISAC to gaze

Transitions
  Next State: 30, Pause and clear ML
    Event: -1, -1


State 70: (Pause and clear ML (ISAC should gaze))
Timeout: 1000, Period:100

Activities
  Trigger 0: ArmLink(0,3), Reset Cmd and Arb vectors

Transitions
  Next State: 80, Load x,y,z information to release Barney
    Event: -1, -1


State 80: (Load x,y,z information to release Barney)
Timeout: 200, Period:100

Activities
  Trigger 0: VSVQ1(1,0), Pass x,y,z data from VS to VQ

Transitions
  Next State: 81, Generate the release trajectory
    Event: -1, -1


State 81: (Generate the release trajectory)
Timeout: 200, Period:100

Activities
  Trigger 0: Gen_Traj_1(0,0), Generate the 3rd trajectory in which
Barney is released
  Trigger 1: ArmLink(3,0), Reset Cmd and Arb vectors

Transitions
  Next State: 82, Initialize the trajectory player
    Event: -1, -1
```

```
State 82: (Initialize the trajectory player)
Timeout: 200, Period:100

Activities
  Trigger 0: Play_Traj_1(1,0), Initialize the trajectory player

Transitions
  Next State: 83, Play the trajectory file to release Barney
    Event: -1, -1


State 83: (Play the trajectory)
Timeout: 3100, Period:1000

Activities
  Trigger 0: Play_Traj_1(0,0), Play the trajectory

Transitions
  Next State: 90, Load data to go to home position
    Event: -1, -1


State 84: (Open hand)
Timeout: 1000, Period:100

Activities
  Trigger 0: CmdCom(2,1), Throw event to RightHand agent to open the
hand

Transitions
  Next State: 90, Open hand
    Event: -1, -1


State 90: (Load x,y,z information to go home)
Timeout: 200, Period:100

Activities
  Trigger 0: Home_Loc(2,0), Load the x,y,z location where Barney is to
be released

Transitions
  Next State: 91, Pass data from VS to VQ
    Event: -1, -1


State 91: (Load the data from the VS to the VQ)
Timeout: 1000, Period:100

Activities
  Trigger 0: VSVQ3(1,0), Pass data from VS to VQ

Transitions
  Next State: 92, Generate the release trajectory
    Event: -1, -1
```

```
State 92: (Generate the release motion trajectory)
Timeout: 500, Period:100

Activities
  Trigger 0: Gen_Traj_3(0,0), Generate the release motion trajectory
  Trigger 1: ArmLink(3,0), Reset the Cmd and Arb vectors

Transitions
  Next State: 93, Initialize the trajectory player
    Event: -1, -1


State 93: (Initialize the trajectory player)
Timeout: 100, Period:100

Activities
  Trigger 0: Play_Traj_3(1,0), Initialize the trajectory player

Transitions
  Next State: 94, Throw event 300,300 to play trajectory and release
Barney
    Event: -1, -1


State 94: (Play trajectory and release Barney)
Timeout: 6000, Period:100

Activities
  Trigger 0: Play_Traj_3(0,0), Play trajectory and release Barney

Transitions
  Next State: 0, Open the hand and finish the demo
    Event: -1, -1
```

APPENDIX B

DEPTH ESTIMATION

Finding the Cartesian coordinates and the depth of a given object given stereoscopic vision is an invaluable resource to allow other parts of the robot to interact with the environment. In particular, a grasp-reach behavior is only possible if the Cartesian coordinates in space are available.

The diagram shown below represents the camera head structure. The left and the right camera are placed on a base of length - *a*. The left camera and the right camera make angles θ_L and θ_R with the horizontal respectively. The goal target is shown as – *P* and the depth to the target is represented by – *d*.

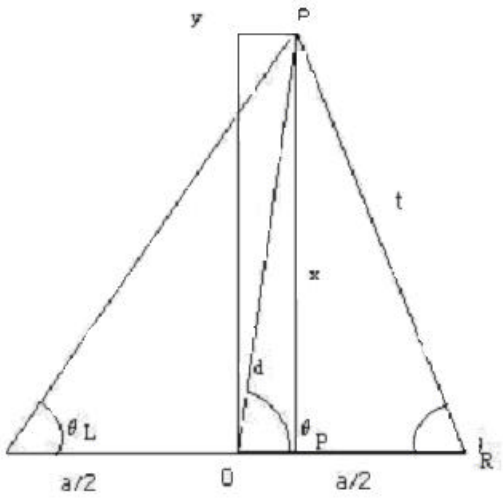

Figure 32. Camera Head Structure.

To find *d*, the Cosine Law was used:

$$d^2 = t^2 + \left(\frac{a}{2}\right)^2 - 2 * t * \left(\frac{a}{2}\right) * Cosq\,R \tag{B.1}$$

However, *t* is unknown and the Sine Law was used to find it:

$$\frac{t}{Sinq\,L} = \frac{a}{Sin(180 - q\,L - q\,R)}$$
$$t = \frac{a * Sinq\,L}{Sin(q\,L + q\,R)} \tag{B.2}$$

Since two separate and independent Pan-Tilt units were used, a single tilt value is computed by finding the average of both tilt values. Since there is an element of height involved in the model, the depth increases proportionally to the tilt. Pythagoras Theorem is used to find the new depth – *d_t*:

$$d\_t^2 = d^2 + h^2 \tag{B.3}$$

Similarly, the Cartesian coordinates of a goal target (assumed to be a point in space) were computed based on figure 33.

The established set of coordinates has *X* coming out of the plane, *Y*, horizontal to the plane, and *Z*, vertical to the plane.

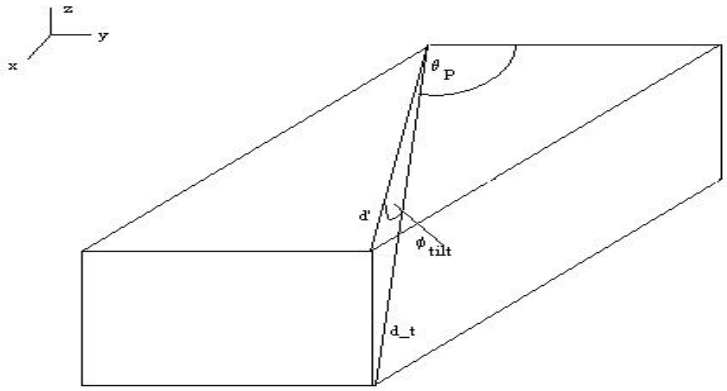

Figure 33. Cartesian Coordinate Model.

From the above diagram, we can obtain the following equations:

$$d = d\_t * Cosf tilt \tag{B.4}$$

$$z = d\_t * Sinf tilt \tag{B.5}$$

Referring to the Camera Head Structure figure, *X* and *Y* are:

$$x = t \sin q R \tag{B.6}$$

$$y = tCosq R \tag{B.7}$$

In this way, the Cartesian coordinates and the depth of the target are computed by the Depth Estimation component. The data is available to any receiving agent.

APPENDIX C

CONVERSION OF A SINGULAR PAN ANGLE

The ISAC humanoid's head structure was modified recently. Previously, a singular pan motion controlled both color cameras. At the time, any agents that needed to communicate with the Head Agent had to do so through a singular pan angle. After the modifications took place, two independent pan-tilt units were operating as the head. Thus, a conversion from a singular pan angle to two pan angles had to be computed.

The model used to develop the computations is based on the following diagram:

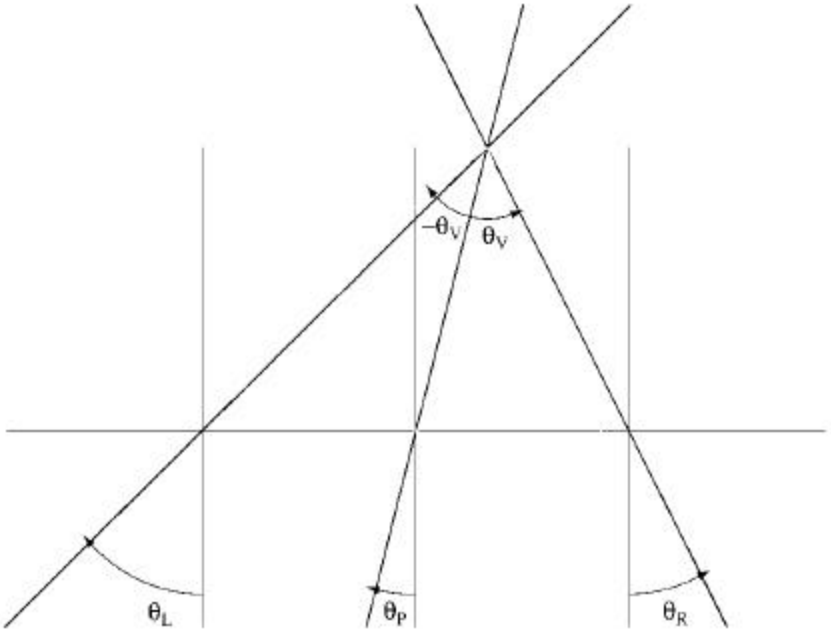

Figure 34. Singular Pan Angle Conversion

Note: Clockwise rotations represent negative angles and counterclockwise rotations represent positive angles.

Based on this model, you can get θ_R and θ_L:

$$q\,R = \arctan[\left(\frac{CosqVSinq\,P + SinqV}{CosqVCosq\,P}\right)] \tag{C.1}$$

$$q\,L = \arctan[\left(\frac{CosqVSinq\,P - SinqV}{CosqVCosq\,P}\right)] \tag{C.2}$$

Solving for the pan angle, the following equations results:

$$q\,p = \arctan[\frac{1}{2}(\tan q\,L + \tan q\,R)] \tag{C.3}$$

$?_P$ is given by the agent, but $?_V$ has to be predetermined. The value was calculated empirically and established at 15 degrees.

This calculation was implemented as an IMA mechanism and is used any time an agent reports to the head with a singular pan angle.

Finally, as a precautionary note, the hardware structure requires a switch between $?_L$ and $?_R$, since the left and right sides of the robot were given when looking towards the robot, not away from it.

APPENDIX D

COM AND DCOM

The following description is based on Olivares overview [2003]. The IMA software platform uses the Component-Object Model (COM) and Distributed COM (DCOM) to provide communication between agents. COM allows objects to interact even if they are running in different computers. Objects can be linked together without the need to compile. Hence, developers do not worry about communication issues.

COM's protocol allows compiled objects to operate across process boundaries without accessing the source code. By using COM and DCOM, a developer writes objects that can be linked to other objects at run time – even with objects in other computers. The communication is possible because of COM's compilers, which place header-like information in the resulting binary files. At runtime, COM sends function calls to their appropriate process by using Microsoft Remote Procedure Calls (RPC). COM does not significantly affect performance on the system.

A proficient and detailed description can be found under Microsoft's Developers Network website at: http://msdn.microsoft.com/library/default.asp.

BIBLIOGRAPHY

Alford, A., M. Cambron, A. Srikaew. 1999. ISAC Operation Manual, *CIS Technical Report*, Vanderbilt Univ., Nashville, TN.

Ao, X. Navigation via Sensory Motor Coordination. 2003. *Master's Thesis*, Vanderbilt Univ., Nashville, TN.

Barile, J. B. 1997. Color Segmentation for Focus of Attention in an Active Trinocular Camera Head. *Master's Thesis*, Vanderbilt Univ., Nashville, TN.

Cambron, M., and R.A. Peters II. 2000. Sensory Motor Control for Grasping in a Humanoid Robot. *Proceedings 2000 IEEE International Conference on Systems, Man and Cybernetics*. Nashville, TN.

Campbell, C. L. 2003. Learning through Teleoperation on Robonaut. *Master's Thesis*, Vanderbilt Univ., Nashville, TN.

Christopher, J.L. 1998. A PneuHand for Human-like Grasping on a Humanoid Robot. *Master's Thesis*, Vanderbilt Univ., Nashville, TN.

Crawford, L. S., 1998. Learning Control of Complex Skills. *Ph. D. Dissertation,* Univ. of California at Berkley.

Daerden, F., D. Lefeber. 2002. Pneumatic Artificial Muscles: Actuators for Robotics and Automation", *European Journal of Mechanical and Environmental Engineering*, 47(1): 10–21.

Direct Perception. 2004. *http://www.dperception.com/products.html#PTU-D46-17*

Falanders, M., J. J. Pellegrini, and S. D. Geisler. 1996. Basic features of phasic activation for reaching in vertical planes. *Experimental Brain Research,* 110, pp. 67-79.

Fitzpatrick, P., G. Metta, L. Natale, S. Rao and G. Sandini. 2003. Learning About Objects Through Action - Initial Steps Towards Artificial Cognition. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Taipei, Taiwan.

Irie, R. E. 1995. Robust Sound Localization: An Application of an Auditory Perception System for a Humanoid Robot. *Master's Thesis*, MIT, Cambridge, MA.

Kawamura, K., C.A. Clifton, K.A. Hambuchen, and P. Ratanaswasd. 2004. MultiAgent-Based Cognitive Robot Architecture and its Realization: Towards Body-Mind Integration. *Submitted to IEEE International Conference on Robotics and Automation (ICRA).*

Kawamura, K., T. E. Rogers, and K.A. Hambuchen. 2002. Towards a Human-Robot Symbiotic System. Invited Paper, *International Manufacturing Automation Leaders Forum (IMLF),* Adelaide, Australia.

Kawamura, K., R. A. Peters II, S. Bagchi, M. Iskarous, and M. Bishay. 1995. Intelligent robotic systems in service of the disabled. *IEEE Transactions on Rehabilitation Engineering,* 3.1:14-21.

Klute, G., B. Hannaford, J. M. Czemiecki. 1999. McKibben Artificial Muscles: Pneumatic Actuators with Biomechanical Intelligence. *Proceedings of the International Conference on Advanced Intelligent Mechatronics,* Atlanta, Georgia.

Klute, G., B. Hannaford. 1998. Fatigue Characteristics of McKibben Artificial Muscle Actuators. *Proceedings IROS*, pp. 1776-82, B. C., Canada.

MXL V57M Large Diaphragm Condenser Microphone. 2004. *http://www.musician-center.com/live/c/Condenser_Mics/MXL_V57M_Large_Diaph_273160.htm.*

Lamb, M. E., M. H. Bornstein, D. M. Teti. 2002. Development in Infancy: An Introduction. *Mahwah, N.J. Lawrence Erlbaum Associates, Inc*., pp. 25-30.

Northrup, S., N. Sarkar, and K. Kawamura. 2001. Biologically-Inspired Control Architecture for a Humanoid Robot. *IEEE International Conference on IROS*, Hawaii.

Pack, R.T. 2002. IMA: The Intelligent Machine Architecture. *Ph. D. dissertation.* Vanderbilt Univ., Nashville, TN, pp. 51-80.

Pack, R.T., D. M. Wilkes, and K. Kawamura. 1997. A Software Architecture for Integrated Service Robot Development. *IEEE Conf. On Systems, Man, and Cybernetics*, Orlando, Fl, pp. 3774-3779.

Peters II, R. A., C. Campbell, R. Bodenheimer. 2004. Superpositioning of Behaviors Learned through Teleoperation. *Submitted to IEEE International Conference on Robotics and Automation (ICRA).*

Pfeifer, R., and C. Scheier. 1997. Sensory-motor coordination: the metaphor and beyond. *Robotics and Autonomous Systems, Special Issue on "Practice and Future of Autonomous Agents,* 20.2-4, pp. 157-178.

Olivares, R. 2003. Intelligent Machine Architecture. Introduction and System Overview. *Intelligent Robotics Laboratory – Internal Technical Document.* Nashville, TN.

Rogers, T. E. 2003. The Human Agent: A Model for Human-Robot Interaction. *Ph. D. dissertation,* Vanderbilt Univ., Nashville, TN.

Scassellati, B. 2000. Theory of Mind for a Humanoid Robot. *First IEEE/RSJ International Conference on Humanoid Robotics*.

Shadow Air Muscles. 2004. http://www.shadow.org.uk/products/airmuscles.shtml#Anchor-Advantages-36621

Shadow Dexterous Hand. 2004. http://www.shadow.org.uk/products/newhand.shtml

Spectral Power Distribution. 2004. *http://hyperphysics.phy-astr.gsu.edu/hbase/vision/spd.html*

Sony XC 999 ultra-compact integrated camera module. 2004. *http://www.avsupply.com/details/xc999.html*.

Srikaew, A. 2000. A Biologically Inspired Active Vision Gaze Controller. *Ph.D. Dissertation*, Vanderbilt Univ.

Yamazaki, Y., H. Itoh, and T. Ohkuwa. 1995. Muscle Activation in the Elbow-Forearm Complex During Rapid Elbow Extension. *Brain Research Bulletin*, 38, pp. 285-295.