ORGANIZATION AND PRIORITIZATION OF SENSORY INFORMATION USING AN EGO-CENTERED, LOCALLY-CONNECTED NETWORK

By

KATHERINE ACHIM FLEMING

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

May, 2009

Nashville, Tennessee

ACKNOWLEDGEMENTS

This work would not have been possible without the financial support provided by a NASA Graduate Student Research Fellowship (GSRP).

I would like to begin by thanking my advisor, Dr. Alan Peters, for his insight, support, and guidance. Thank you for your encouragement and faith in my abilities during my time at Vanderbilt.

I am also deeply grateful to the other members of my dissertation committee. Many thanks to Dr. Bobby Bodenheimer for taking the time to guide me through this process; your help has made me a much better researcher and presenter. Thanks you to Dr. Kazuhiko Kawamura and Dr. Mitch Wilkes for all of your advice and insights during this process. Thank you also to my outside committee members, Dr. Bill Smart and Dr. Kim Hambuchen; your input and guidance have been much appreciated.

A huge thanks to Flo Wahidi, who keeps the CIS running smoothly. I do not even want to imagine this experience without her. Thanks also to my fellow students—past and present—in the CIS; your camaraderie and moral support has been much appreciated.

I would like to thank my family as well as all of my friends for their endless love and support. This would not have been possible without your encouragements. Finally, a million thanks to Paul: j'ai fini moi aussi!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

This dissertation describes a multimodal attention system for a sensor-guided robot. The robot has a number of different sensors that continually send it information about the environment and itself. To perform a task successfully, the robot must organize that information, pay attention to that which is important, and ignore the rest – unless something unexpected and potentially beneficial or dangerous occurs. Then the robot must shift its attention to the new stimulus, assess its importance, and respond accordingly. The ability to "pay attention" to sensory stimuli of multiple types (sight, sound, touch, smell, taste, etc.) and to respond appropriately is shared by humans and other mammals [4]. Some of the underlying neural mechanisms are well understood in terms of their common attributes if not their higher-level interactions. In particular, most animal attentional mechanisms incorporate sensitization – a priming for anticipation of an event, and habituation – the ability to ignore a benign or useless stimulus [5]. Attentional systems for robots are being studied and developed, but mostly these deal with a single sensory mode and few exhibit sensitization and habituation. The attentional system described here can process multiple sensory modalities, and exhibits both the characteristic responses.

The system is implemented on an egocentric, conformal network of nodes called a Sensory Ego-Sphere (SES) [1]. The SES is centered on the robot's base frame of reference. The collection of nodes partitions space into polygonal solid angular regions that emanate from the base frame. Each node continually receives, from parallel concurrent sensory processing modules (SPMs), all sensory data derived from one such region. Along with this sensory data, the SPM sends an estimate of its angular position and the time at which the data was received. Adjacent nodes receive from adjacent regions. Also associated with each node is a list of activation values that represent the salience of the data with respect to the robot's current task. A node interacts with its neighbors to adjust both its current activation values and the change in activation with respect to its previous values. The aggregate of these two variables over all the nodes directs a global focus of attention (FOA) across the SES and, therefore, the robot's locale. Because the robot's sensors collect a large amount of data, most of which is irrelevant to the task, the good selection of a single location to which to attend conserves the robot's resources. Thus, in general the robot directs its task-oriented resources to the region of space that contains the node with the focus of attention. If that focus is entirely task-driven, however, important changes in the environment could be missed. If one or more sensors detect something new or detect a significant change in stimulus, a change in the focus of attention may be called for, at least momentarily. By making each node a simple computational object that acts independently of the others, the net-like interconnectivity of the SES enables the nodes' local interactions to produce a global shift in attention. This makes use of the two activation states, absolute and differential, and a prioritized list of tasks.

Within this system, the attentional aspects of a task are defined in terms of the sensory modalities that comprise them. Task priority is predetermined. The system is preemptive so that unexpected changes in stimuli can be attended to, even if their attributes do not match those of the currently active task. If perchance the new stimuli match a task of higher priority than the current one, the higher-priority task is initiated until the situation resolves, at which point the original task resumes.

Although task preemption is an important functionality of the system it has the drawback that attention shifts can occur to locations of neither importance to the task nor to the safety of the robot or environs. Overall performance decreases when the stimulus causing a FOA shift does not elicit a higher-priority task. Such stimuli are considered to be *distractors*. Habituation mechanisms mitigate the decline in performance by enabling the system to ignore irrelevant stimuli that reoccur. Shifts to such distractor locations become less frequent.

In theory, the system is designed for a robot that operates in a dynamic environment in real-time. The parallel implementation of sensory processors and of the nodes as computational objects could make this possible (as could dramatic increases in the speed of serial computation). The effective connectivity of the nodes, the methods of node communication, and the sensory processors are suitable for use in a distributed system. At this point in time, however, such an actual or quasi-distributed implementation of the SES has not been implemented. Apart from the real-time collection of sensory data, the computations for this work were performed off line on serial machines to provide a proof of concept. Although the SES can be used on stationary as well as mobile robots, motion transforms (available in [1]) must be applied when the robot is moving around the locale.

Why Pay Attention?

The field of robotics has demonstrated success in industrial automation, where robots operate in predictable environments. Industrial robots usually have a limited number of well-defined tasks to perform, limited sensory inputs (if any at all), and controllers that, designed for preprogrammed point-to-point motion, have no extra computational resources for the analysis of sensory information. Industrial robots are seldom mobile and generally not used in situations where interaction with people is possible. Nevertheless they have been enormously successful for the automation of manufacturing [6]. With a few notable exceptions—like the iRobot©Roomba©[7], a vacuum cleaning robot—robots are not deployed in unpredictable, dynamic environments where people are in close proximity. Consequently, much current research is on the design and implementation of reactive, multi-functional robots that can be so used. Such robots require an increase in their versatility – robots should be capable of completing a variety of tasks as opposed to one explicitly-defined task – as well as adaptability – robots should have the ability to handle unexpected events and safety concerns while completing tasks. As these capabilities are achieved, robots are likely to be used for applications such as office delivery, warehouse operations (moving crates, restocking), search and rescue, and increasingly complex space exploration missions.

Over the past few decades, the steady increase in power and simultaneous decrease in cost of computation [8] has made the realtime extraction of useful information from sensory signals much more feasible. Couple that with recent decreases in the cost of sensors (most notably cameras [9]) and the reality of intelligent robots is more likely than ever. The challenge is for the robot determine with reasonable accuracy which of the information it receives from the torrent of sensory data is significant. It is necessary to exclude spurious data but that requires either knowing which data are spurious or knowing which are necessary. In animals, attention appears to serve the purpose of stabilizing or tracking an aggregate of sensory information once it is identified as useful so that the information may be acted upon purposefully. Of course, it could also be that attention is *the result* of the filtering of spurious information (that which remains when all else is excluded) [4].

The system briefly described above, and analyzed in this dissertation, incorporates both ideas. A task description specifies cues; *i.e.*, relevant sensory information that is to be found and attended to. The spatially-distributed computational nodes operate in parallel on data supplied by directional sensors. Each node computes for each sensory modality an activation value that is a measure of similarity between the cue and the data in its region. The node computes a total activation from those of the individual modalities. The node with the greatest activation emerges as a candidate for the focus of attention. The parallel processing at once finds the cue and suppresses spurious information. The resultant node points to the region that is *a candidate* for FOA but not necessarily *the* FOA. A system that doggedly tracks the maximum activation will often miss new data that might indicate an event more important than the current task. But, then again, it might not. Thus the system incorporates both sensitization and habituation. The result is a single Focus of Attention that allocates computational resources toward task-significant locations in space, that switches to explore significant new events, but does not persist in such switching if the event is irrelevant.

While numerous techniques for machine vision and attention have been researched, lacking is a comprehensive model that leverages their interrelationships not only to provide a coherent and efficient computational implementation but also to foster affordances that emerge from their structured combination.

An attentional network on the SES was previously developed by Hambuchen [10]. Attention was directed to sensory events bound in space and time. Events were extracted from data gathered from a robot's sensors and were registered to the SES. Salience was assigned not only to events related to the task, but also to locations corresponding to multiple co-occurring events. Salience calculations were based on the occurrence of an event, the relevance of the event to the task, and whether the event is habitual. Repetitive events occurring within a fixed interval set *a priori* were habituated over time.

Contribution of this Work

The attention system for robots described in this dissertation is multimodal over an arbitrary, dynamic set of sensor modalities. That is, the system is designed to work simultaneously with any number of different sensory modalities and is fully independent of sensor type or abstracted sensory information, providing that the information is spatially distributed. The number and types of sensors defined in the system can vary. Although implemented on a pre-existing short-term memory structure, the SES [1], which has been used as an attentional network in the past by [10], the system described here extends this work in several novel ways.

A FOA emerges from the aggregated behavior of a spatially distributed set of locally-connected, simple, reactive processing nodes that are cued by a global descriptor that lists sensory characteristics relevant to the task at hand. This simultaneously suppresses information of little importance. This system retains the original sensory data, processes it all, and registers it to the SES, as opposed to registering only abstracted sensory events extracted from the original data.

The system is designed to simulate each node having independent processing power and only communicating with its immediate neighbors; this was done to facilitate a parallel implementation in the future. For this purpose, distributed schemes of spreading sensory data as well as their task-related activation were developed based on the serial implementation presented in [10]. A distributed scheme for spreading habituation effects was also developed.

The FOA selects a direction in space based on both the absolute salience to the task of the sensory stimuli in that direction, and on the first-order temporal dynamics of the stream of incoming sensory data. That is, the FOA is sensitized toward task-relevant data and toward abrupt changes in the sensory background. The dynamics of the response of the individual nodes to the local sensory data stream cause the FOA to habituate to task-irrelevant and benign, repetitive stimuli. In the previous implementation of [10], FOA selection was based on the number of events at a location, the task-relevance of the events, and their time of registration onto the SES.

In [10], all repetitive stimuli was habituated on a fixed interval length determined *a priori*, regardless of task-relevance. In this system, only task-irrelevant stimuli are habituated. Habituation is applied to any stimuli that attracted the FOA yet does not match a higher-priority task, regardless of its presentation interval length, whether periodic or constant. Habituation to a specific stimulus decays over time so that the

FOA can shift back to it if the stimulus suddenly appears or disappears later. This set of behaviors provides a robot with the ability to attend to a task without repeated distraction while preserving the ability to attend to novel stimuli. Such abilities are essential for autonomous behavior in loosely-structured environments and enable the failsafe response to danger.

Overview

The remainder of this dissertation is organized as follows. Chapter II includes an overview of the Sensory Ego-Sphere. A review of the literature on previous works in attention and habituation, as well as its relation to the system described in this dissertation is also found in this chapter. Chapter III first presents the overall attention system, followed by a more detailed description of its local and global activity levels. A description of the habituation mechanisms implemented in the system concludes this chapter. Chapter IV presents all experiments performed in this work as well as a discussion of the results. Finally, chapter V contains conclusions and recommendations for future directions of study.

CHAPTER II

BACKGROUND AND RELATED WORK

The attentional network described here is implemented on a Sensory Ego-Sphere, incorporates ideas from studies of attention in humans and other animals, and exhibits both sensitization and habituation. Hence these three topics are discussed below.

Sensory Ego-Sphere

The Sensory Ego-Sphere (SES) can be thought of as a mediating interface between sensors and cognition. It is an egocentric short-term memory, and it structures, stores, and coordinates multimodal sensory information for further processing. It was designed to mimic some of the functions performed by the mammalian hippocampus, such as the integration of multiple streams of sensory data and integration of sensory and motor information [11]. It can provide an egocentric mapping of the environment as well as information about position (of self) with respect to objects in the environment. [1]

The SES is a virtual tessellated sphere centered at the robot's base frame, and its orientation remains fixed with respect to the world (figure 1). As a robot moves, its heading changes on the SES, and data can be moved from node to node based on the robot's movements. The SES thus simplifies the organization, storage, and retrieval of egocentric information. [1]

The tessellation used in the SES is geodesic and partitions space into a set of hexagonal or pentagonal cones that emanate from the frame. A geodesic dome is composed of twelve pentagons and a variable number of hexagons that depend on the frequency (or tessellation) of the dome. The frequency is determined by the number of vertices that connect the center of one pentagon to the center of another pentagon,

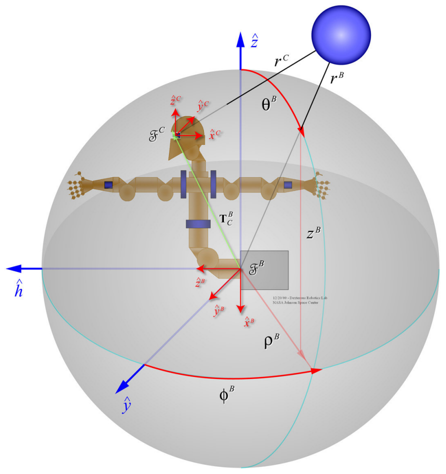


Figure 1: A robot within its SES [1].

all pentagons being distributed on the dome evenly, as shown in figure 3. The number of vertices (V) can be determined from the frequency (N) using equation 1.

$$V = 10N^2 + 2 \tag{1}$$

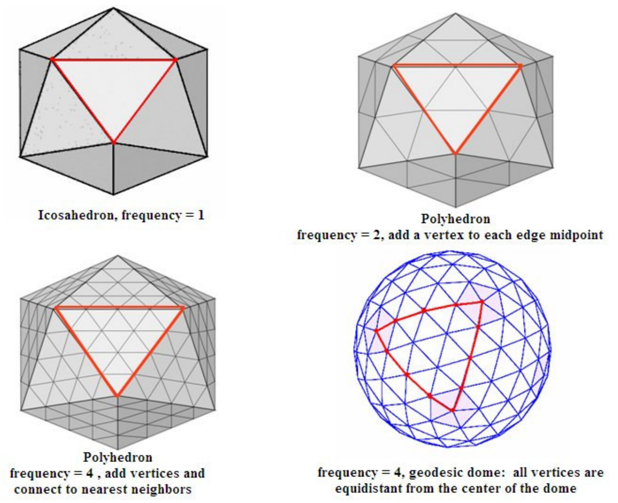

Icosahedron, frequency = 1
Polyhedron
frequency = 2, add a vertex to each edge midpoint
Polyhedron
frequency = 4 , add vertices and connect to nearest neighbors
frequency = 4, geodesic dome: all vertices are equidistant from the center of the dome

Figure 2: Tessellation of an Icosahedron into a Geodesic Dome [2].

Figure 2 illustrates the creation of a geodesic dome. First, an icosahedron is created; this dome has a frequency of 1 and is made up of 12 pentagons. A new vertex is added at the midpoint of each edge, increasing the frequency of the dome to two. Each new vertex is connected to the two vertices on the original edge as well as to the four new vertices nearest it, resulting in a total of six neighbors for each new vertex. The original pentagon centers are connected to five neighbors while the new hexagon centers are connected to six neighbors. Subdivision continues until the polyhedron has the desired frequency. Once all vertices have been added to the polyhedron structure, the polyhedron is reshaped so that all vertices are equidistant from the center, creating the geodesic dome. [1]

Each vertex in the tessellation corresponds to a node where sensory and motor data related to the region can be stored. Nodes at the center of a hexagonal region have six neighbors while nodes inside a pentagonal region have 5. A tessellation frequency of $N = 14$ is used in this work, which partitions space into 1963 regions such that the angle between nodes is on the order of $4.5° - 5°$. For example, an object that has been visually identified in the environment is projected onto the sphere at azimuth and elevation angles that correspond to its location with respect to the SES frame. A label that identifies the object and other relevant information is then stored. The vertex on the sphere closest to an object's projection becomes the registration node, as illustrated in figure 3.

The geodesic dome structure was chosen to represent the SES because it is "the optimal solution to the problem of how to cover a sphere with the least number of partially overlapping circles of the same radius" [12].

Sparse and/or simple sensory modalities such as sound or IR motion detection can be combined accurately using an SES and attention can be cued by such combinations [1]. More specifically, sensory information having the same source (for example, the sound of the voice and the image of the face of a person sensed simultaneously from the same direction in space) can be recognized as such through spatio-temporal

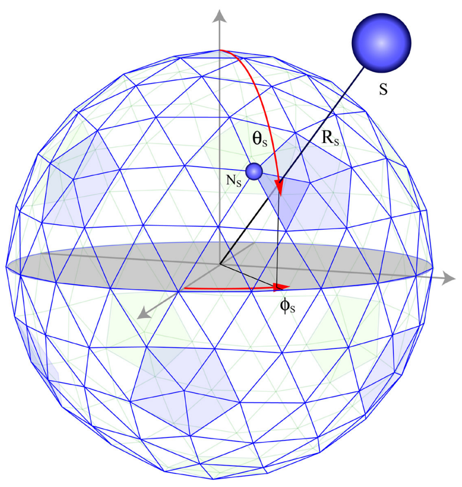


Figure 3: Projection of an object onto the SES [1].

coincidence of stimuli, and can increase the salience of a particular location on the SES. Various sensory information emanating from the same location in space but distributed over time can also be accumulated to form a focus of attention on the sphere [10]. The SES also possesses sensitization (increasing the salience of an unexpected event) and habituation (decreasing the salience of events that have periodic occurrences) capabilities with respect to attention [10].

Since the SES also serves as a short-term memory, it can keep track of objects in the robot's environment. This can be used to provide a display of the robot's locale-specific knowledge that can be of use to supervisors or remote operators of the robot as well as persons who interact with the robot [13, 14]. The SES has also been used on at least one mobile platform to perform spatial localization of the robot and navigation in 2D [15].

An object's spatial distribution on the SES over time can also be tracked [16]. The accumulated information can then be used to identify the most likely locations for a specific object type to appear, which can be used as a starting point for a visual search.

The SES is not limited to the storage of sparse sensory information; it can also store images with higher resolution than the SES itself. Spatially-overlapping imagery can be stored in the SES database and a spherical composite of its visual contents can be generated and updated [2]. This composite image is both a map of the locale and a representation of the local contents of the underlying full-resolution imagery. Visual attention methods have been used to indicate areas in the environment where the robot may need to apply its cognitive resources [2]. It also enables fast alignment of overlapping images without warping or position optimization, since an attentional point (AP) on the composite typically corresponds to one on each of the collocated regions in the images [17]. Such alignment speeds analysis of the multiple images of the area.

Attention

Attention is the process of selecting one object to concentrate on while ignoring the others. Attention is important because humans and robots alike do not have the computational capabilities to process all available information instantaneously. It guides the exploration of the locale toward relevant locations—locations that contain useful knowledge pertaining to the current task. It is essential to filter out irrelevant information so that the computational resources can be assigned solely to the attended object for further analysis, representation, or behavioral control [18]. While attention filters out information not relevant to the task at hand, it can shift its location in response to a changing or unexpected situation, or to focus on another aspect of the task at hand. The attention system must balance task completion against potentially dangerous events: unexpected sensory information must be attended to avoid unsafe situations (sensitization), and this information can be ignored at future times if it is deemed safe or becomes repetitive (habituation). Attention is then a function of multimodal sensory information in space and time; it is driven by the data and modulated by the task, and can be interrupted based on low-level reflexes of safety.

Attention has been a widely-studied topic in psychology and cognitive neuroscience, dating as far back as 1890 [19], and is still being investigated today. Recently, studies using neuroimaging techniques such as PET [20] and fMRI [21] have identified areas of the brain modulated by attention: regions in the frontal, occipital, parietal, and primary visual cortices as well as the superior colliculus, to name a few. Because the attention literature is so extensive, the following review centers on attentional models developed for robotic applications. For more information on the human attention system, see [4].

Attention can be divided into two separate mechanisms, bottom-up and top-down; these mechanisms have different methods of calculating the "salience" of a location. The bottom-up mechanism defines salience as conspicuousness: the more a location's features—color, shape, orientation, movement, etc.—locally stand out from those of its neighborhood, the more it is salient. An example of this is how a yellow banana stands out in a bowl filled with red fruits. For the top-down mechanism, salience is determined by the task at hand: the banana in a bowl of red fruits would not be salient if we are specifically searching for a red strawberry.

In addition to the bottom-up and top-down classification of attention models, attention systems can be categorized in a number of ways: some models deal solely with visual attention while others incorporate different sensory modalities; certain models concentrate on a single sensor and others are multimodal. Finally, some models are designed to generate results on static images while others operate in dynamic applications. I have attempted to make these distinctions in the following discussion without a full categorization of each attention system.

Feature-Integration Theory

An influential contribution to attention research is Treisman and Gelade's feature-integration theory of attention [22], which states that separable features such as color, orientation, and brightness, are automatically registered in parallel. However, attention is necessary to bind all of these features observed at one location together into an object—attention is directed at locations where features come together—, and this binding process is serial. Several computational models of visual attention have been developed based of this theory [23, 24, 25]; these models create an independent map for each type of feature used and combine these into an overall "saliency map" [26], where each location in the map contains a value reflecting the visual conspicuousness of the corresponding location. The saliency map is then used to direct the focus of attention by selecting the location with the highest value. Color, orientation, and luminance are the most frequently-used low-level features in the models discussed below.

Itti et al. [23] have developed a popular model of visual attention. This bottom-up model is made up of parallel feature maps—6 maps for intensity contrasts, 12 for color discrimination, and 24 maps for orientation discrimination, all computed at different resolutions. Each feature's salience is computed using a center-surround mechanism similar to the process taking place in human visual receptive fields, where a certain stimulus must be "on" in the center of the receptive field and "off" in the surround, or vice-versa. This process is implemented through the difference between a high-resolution pixel location and its lower-resolution surround, to identify locations which stand out from their local neighborhood. The feature maps are then combined first into conspicuity maps for intensity, color, and orientation respectively; these three maps are then combined linearly into a single saliency map from which the maximally salient location can be obtained; this location becomes the focus of attention. An inhibition of return mechanism is also implemented in the event that the most salient location is not the correct location; this location would be inhibited for a fixed time period so that other locations can be attended. A top-down component for the inclusion of task-relevance information was later added to this model; this component biases the system for those features present in the target [27]. This algorithm was also combined with an object recognition model to identify attended objects [28].

Another feature map-based visual attention model is Cave's FeatureGate model [24]. In addition to having a corresponding value in whatever feature maps are defined in the system, each location in the visual scene has an attentional gate to regulate the flow of information from that location's value to the output based on the features of the location and those of its neighbors. This is a hierarchical model where each location competes to be the focus of attention; the winner of each neighborhood goes on to the next level until a single location remains. This model has both a bottom-up and a top-down subsystem: the bottom-up process computes salience values and identifies the most conspicuous locations independently of the task while the top-down process will open the gates of those locations similar to the target and close the gates of those with features unlike the target. The salience from the bottom-up process and discrimination values from the top-down process are then combined into a single activation map and the top activations in that map are passed to the next level, where the process is repeated. Inhibition of return is also incorporated in this model by reducing the activation of the winning location for a period of time if that location is not the target and repeating the process a few times so that other winners are chosen. This model has been implemented on a humanoid robot and the results were shown to be consistent with the human attention system [29].

Guided Search [25] is another model of visual attention and its results are closely related to human data acquired during visual searches. Its structure resembles FeatureGate: color and orientation feature-maps are used as well as bottom-up and top-down components, which combine to form a single activation map whose peaks represent areas of the scene that are of potential interest. The top location can then have access to higher-level computational resources such as an object recognition module. As in the other models above, inhibition of return is also incorporated into the Guided Search model. Revisions to this model have been made [30, 31, 32] for a more accurate human vision system model. Among the changes were the addition of a size feature map and the incorporation of eye movements.

Among the many attention models that include a saliency map, there is a wide variety of features used to generate the map and different techniques exist to extract relevant information from that map. Breazeal and Scassellati's model [33] is based on Wolfe's work and uses face, motion, color, and habituation feature maps that are modulated by motivations and behaviors in a top-down fashion.

Ude et al. [34, 35] use motion and disparity in addition to the color, orientation, and intensity feature streams. The distributed implementation processes each stream in parallel along with an additional stream, based on the FeatureGate model, to integrate top-down influences .

Maki et al [36] integrate image flow, depth (stereo disparity), and motion while Frintrop et al. [37] use intensity and orientation features from depth and reflectance images obtained from a 3D laser scan to create saliency maps.

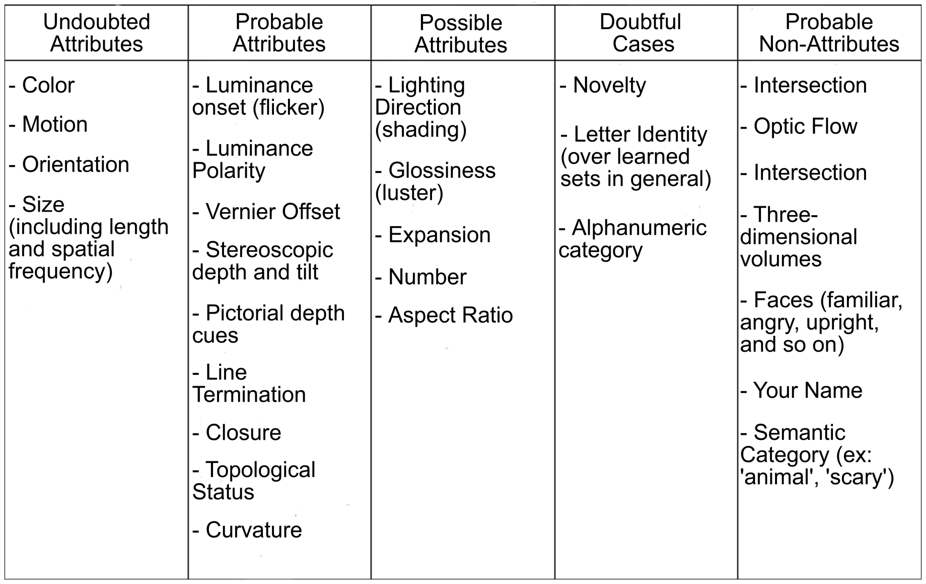
Heidemann [38] creates a saliency map from local color symmetries—obtained from image gradient calculations—to identify focus points (FPs) which correspond to the center of interesting regions in the visual stream. This algorithm was shown to be stable under object rotation, illumination changes, and noise; results were more distinct and meaningful than those generated from algorithms that detect corners and edges [39].

The system by Ma and Zhang [40] uses a contrast-based saliency map to represent color-contrast as well as texture and shape approximation data. A fuzzy growing technique is then used to identify salient points and areas in the saliency map. Lee et al. [41] make use of aspect-ratio, symmetry, and shape features in their system; an Interactive Spiking Neural Network (ISNN) integrates bottom-up and top-down information to identify human faces in the image.

The dynamic visual attention model developed by Backer and Mertsching [42] uses features like color-contrast, edge symmetry and eccentricity, and depth from stereo images in the saliency-map computation and identifies discrete regions of sustained interest with a neural network that performs spatiotemporal integration and local inhibition. Symbolic object files are generated for each discrete region; these files make up the world model and are updated continually. A second stage is used to select a single focus of attention based on task, which is then sent to higher-level processes such as object recognition. Lòpez et al. [43] also developed a dynamic visual attention model that can be used with moving cameras. This model incorporates the shape and motion of the objects to direct attention and uses an attention reinforcement mechanism to maintain focus on objects of interest. Observer commands are used during the process to further guide the deployment of attention.

Although it is not certain which stimuli features guide the deployment of attention, Wolfe and Horowitz [3] have compiled a list of potential attention-guiding features, arranged in categories from unquestionable features to highly unlikely ones. Their list of features is reproduced in table 1.

Table 1: Attributes and likelihood of guiding attention [3]



Undoubted Attributes	Probable Attributes	Possible Attributes	Doubtful Cases	Probable Non-Attributes
- Color - Motion - Orientation - Size (including length and spatial frequency)	- Luminance onset (flicker) - Luminance Polarity - Vernier Offset - Stereoscopic depth and tilt - Pictorial depth cues - Line Termination - Closure - Topological Status - Curvature	- Lighting Direction (shading) - Glossiness (luster) - Expansion - Number - Aspect Ratio	- Novelty - Letter Identity (over learned sets in general) - Alphanumeric category	- Intersection - Optic Flow - Intersection - Three-dimensional volumes - Faces (familiar, angry, upright, and so on) - Your Name - Semantic Category (ex: 'animal', 'scary')

Heidemann's choice of color symmetry was prompted by studies indicating that "symmetry catches the eye"[44], while the special characteristic of depth in conjunctive searches was reported by Nakayama and Silverman [45]. Targets that differ from distractors in only one dimension (color or orientation, for example) can be identified in the same amount of time independently of the number of distractors present—indicating a parallel search; if the target differs in more than one dimension however, a serial search must be performed and the search time is dependent on the number of distractors [22]. Nakayama and Silverman have found that depth is an exception to this rule: if depth is one of the dimensions in the search, another dimension can be searched in parallel. This and the fact that robots need to attend to objects or people in their proximity could indicate that depth is a useful feature for guiding attention.

Biased Competition Hypothesis

The biased competition (BC) hypothesis of Desimone and Duncan [18] builds on another influential idea taken from the psychology literature. This hypothesis states that the competition that objects engage in for resources is biased in favor of task- or behavior-relevant objects; this is done by enhancing the response of relevant features while decreasing the response of neighboring distractors. (The SES is capable of this behavior; see section II.) Attention can be viewed as having two separate modes of operation: spatial attention, which performs in-depth visual analysis on a single location while all other locations are ignored (like a spotlight), and object attention, the process of searching a visual scene for the features of a particular target. Therefore, the top-down competition bias can be applied to either a spatial location or an object's features.

Deco [46] has developed a computational model based on the BC hypothesis in which the spatial and object attention modes are integrated to perform both visual searches and object recognition tasks. Begum et al. [47] have developed a probabilistic model of visual attention, also based on the BC hypothesis, for use on a humanoid robot. A bottom-up competitive component and a top-down modulating component are modeled by probabilistic distributions, which are approximated by a particle filter. The psychological findings that objects located in the fovea generate a higher response than those in the periphery because of varying spatial sensitivity is also modeled: there is a higher transition probability for locations near the focus of attention than for those in the periphery. Dynamically constructed Gaussian Adaptive Resonance Theory (DC-GART) is used to model both working memory and long-term memory; the features and categories of attended stimuli are learned, and the system is biased in favor of new stimuli.

Like Begum et al., Jagersand [48] uses a probabilistic saliency map but this map contains a measure of the amount of information located at a particular location and at a particular scale. This information theoretic approach allows attention to be deployed on relevant data using adequately-sized operators. This saves time and effort as the author observed that, in images of man-made objects, relevant information is found only in a small number of scales. Approaches that consider scale usually test a large variety of different scales without any measure of the amount of information they contain.

Tsotsos' Selective Tuning model [49, 50] is a pyramidal network that incorporates spatial inhibition and task-irrelevance inhibition at each level. The input stimulates each level of the pyramid until the top is reached. A winner-take-all process is then activated at the top level; this is repeated at each lower layer by selecting the strongest location in the winning location's receptive field and inhibiting the non-contributing, spatially-adjacent locations. Experiments were performed with images; it is mentioned that the framework is not limited to this type of input but it appears to be limited to a single input as opposed to the conjunction of more than one type of input [51]. Another network-based implementation is the Selective Attention Identification

Model (SAIM), where the focus of attention (FOA) is selected through a constraint satisfaction process [52].

The attention model developed by Aziz et al. [53] is structured as a list of regions obtained from first segmenting the vision input. Feature values for color contrast, eccentricity, orientation, and symmetry are associated with each region; and the maximum value of each feature gets its turn to selects the FOA. For example, the maximum color contrast value becomes the FOA at $t = 1$; this feature type is then inhibited and the maximum eccentricity value determines the FOA at $t = 2$, and so on.

Multimodal Attention

Of particular interest to our work are dynamic models that include multimodal sensory information to guide attention. The term "multimodal" can be applied to a variety of components of an attention system; a person can guide a robot's attention toward an object with multiple actions—gestures and speech (joint attention) [54]—and separate sensory modalities can be used for the robot's scene processing and the user's commands to the robot. We are most interested in systems that use some functional combination of sensory modalities to identify salient regions in the locale based on conspicuousness and task—since a task frequently has objectives that depend on a specific combination of sensory modalities and their relative importance.

Some multimodal attention systems are application-specific and consider a single task: Wilhelm et al. [55] use vision and sonar information to track faces; both Déniz et al. [56]—who combine vision and sound feature maps to shift the FOA—and Lang et al. [57]—who use camera, microphone, and laser range finder information to anchor the face, legs, and speech of a person and deploy attention—track people who may possibly interact with the robot. These systems have had successful results but are not general enough for the variety of tasks that a humanoid robot could be presented with.

The multimodal system of Frintrop et al. [37] combines depth and reflectance images from 3D laser scans while Ouerhani and Hugli [58] incorporate a depth feature map obtained from a range finder to the intensity, color, and intensity gradients features of Milanese's static model of visual attention [59]. Mean curvature and depth gradient, also acquired from the range finder, may also be considered in a future implementation.

The framework of Koene et al. [60] integrates audio and visual feature maps along with a top-down gating mechanism—both inhibitory and excitatory—derived from FeatureGate into a module that shifts the gaze of a humanoid robot. In the attention system of Haasch et al. [54], the importance of certain objects is communicated to a robot through a user's multimodal actions—such as gestures and speech—, and this process guides attention deployment. The verbal commands are stored with the object's visual characteristics obtained from the camera image; however, no other sensory modalities of the object itself—such as whether the object makes a noise—are used.

Crespo et al. [61] developed AMADIS, a general architecture composed of task-related "attentors", each building over time a probabilistic map of the location of the object it is dedicated to find. These maps are then integrated to create the overall attentional space. Complexity is increased by defining different types of attentors: primary attentors receive sensory inputs only; secondary attentors can receive sensory inputs as well as outputs from primary attentors; and instinctive and reactive attentors are used to detect emergency situations where sudden changes are required. Although this general system is said to be multimodal, the only demonstration made in that regard was of a separate sensory modality (sound) being used to change the task, and therefore the attentor module that guides attention. Attentors involving the combination of various sensory inputs were not presented.

Goncalves [62] integrates visual and haptic information to guide the focus of attention. Attention values are weighted and summed across feature maps; these weights depend on the task and the author is investigating a Q-learning strategy to learn them.

Research has also been done in the area of on-line task-specific feature learning for attentional systems: Rajeandran and Huber [63] developed a system that uses reinforcement learning to learn what features should be attended—from a list of identified features—to perform the current task[1] and Baluja and Pomerleau [64] use a neural network to learn which features to use for a particular task over time and to compute expectation values for future features. Feature-learning is not a part of this work; we assume that a task description in terms of multimodal sensory features is known *a priori*.

Attention mechanisms have been used in applications such as content-based image retrieval [65, 66], social attention [67, 68, 69], object recognition [70], and scene segmentation [36].

The large variety and volume of attention models and algorithms suggests that a standardized evaluation method is needed. Shic and Scassellati [71] propose quantitative methods for performance evaluation of computational models of visual attention in terms of similarity to human performance. Williams and Draper [72] disagree with this notion and believe that attention systems should be directly compared to establish performance baselines. Moreover, human performance can only be compared to models that solely use vision; this would not be relevant to the many attention systems—including the one proposed in this work—that rely on multimodal sensory information.

[1]The number of features associated with a particular task is currently limited to 2, and only vision information is used, although it is mentioned that other sensory modes could be used.

The SES-Based Attention System

In the SES-based attention system described in this work, parallel independent sensory processing modules (SPMs) provide the multimodal sensory inputs and features used to direct attention; these features are modulated by the task and are not limited to vision sensors. In a related SES-based attention system previously implemented by Hambuchen [10], vision, sound, IR, and tactile sensors were used as SPM inputs and SPMs included color segmentation modules, motion detectors, and sound localizers. SPM outputs have been low-density—only recognized objects were registered to the SES and the raw sensory information was lost. The attention system implementation described in this dissertation (chapter III) retains the original raw data, processes it, and registers it to SES.

Previously, SPMs have assigned salience to their outputs at the time of SES registration; salience was determined by incidence, task-relevance, and habituation, and was decayed to 0 over a period of time [10]. Because of resolution differences between sensors as well as the tessellation of the SES, the salience of an event was spread to neighboring nodes according to sensor error and accuracy. The attention network then scanned the SES for areas of high salience; the attentional winner was sent to an event binding process to group together events that occurred at the same time and originated from the same source.

In the current implementation, the salience of different regions in the robot's locale is modulated in a distributed fashion based on local neighborhood interactions.[2] SPMs register sensory events and their preliminary activation/salience onto the SES at a particular node; this node then initiates the spread of information to its immediate neighbors based on sensor resolution and uncertainty. Salience is based on task-relevance and is summed across sensory modalities at each node location. A focus of attention (FOA) selector module determines the most salient location as a

[2]Such a design enables parallel implementation.

function of both the absolute salience and the time-change in salience. This differential function enables the system to exhibit habituation and sensitization, and is unique to this system.

Few parallel or distributed attention schemes were found in the literature. Ouerhani and Hugli [73] have implemented a real-time parallel version of the saliency-based visual attention model on a compact architecture called ProtoEye. Sela and Levine's real-time system [74] is implemented on a network of parallel processors and mainly consists of computing the fovea and periphery regions concurrently. However, both implementations involve vision only and do not distribute the computations in the locally-connected manner that we investigate.

The system developed by Sha'ashua and Ullman [75] is more relevant: they use a locally-connected network to iteratively grow a map of globally-salient and spatially-distributed structures (specifically long, smooth curves). Their work incorporates a measure of structural salience determined by the relative placement of features. For example, their system detects line segments arranged in a circular shape amid randomly oriented line segment distractors. This contrasts to local salience that is a function the difference between a single element and its neighbors with respect to a single feature.

Although several models make provisions for top-down control [23, 29, 25], very few implementations actually include top-down mechanisms. Our implementation combines both top-down and bottom-up control to enable dynamic, multimodal attention. Although some of the systems presented above did have multimodal capabilities, few were general in both the number and type of sensors used and the number of tasks that could be performed. The attention system presented here does not depend on the number and type of sensors used.

If a robot is to operate in close proximity to people and/or in an unpredictable environment, safety is of prime importance. The robot must quickly detect unsafe situations and react to them. An exclusive focus of attention would tend to subvert such reactions. That is the robot must be interruptible by safety routines to avoid potentially dangerous situations if possible. Schlosser and Kroschel [76] propose dedicated mechanisms to separate from a robot's attention system to detect potentially dangerous situations such as objects falling down, objects moving rapidly, and humans present in the vicinity of the robot. Such a dedicated mechanism must be designed to interact with an attention system or override it.

It is possible, however, to incorporate safety within the attention system itself. Consider an attention system that (1) is sensitized to characteristic features (*e.g.* a rapid motion, a loud sound, light of a particular color), (2) sensitized to unexpected changes (the onset or terminus of features) and that, (3) concurrently computes saliences at a set of spatially distributed localities both at and away from the FOA. Such a system can react to distal events and shift the FOA there so that the robot's computational resources can be deployed to evaluate the new stimuli. Such an attentional system is a safety mechanism intrinsically.

The problem with a purely sensitized system is that the FOA may shift incessantly to features that are simple distractors, of no importance to the robot's task and of no danger to the robot, its peers, or the environment. Habituation counters that tendency by temporarily suppressing the salience of distractors. Unlike any of the other proposed attention systems for robots, ours incorporates habituation.

Habituation

Habituation is a process that modulates the shifting of the focus of attention. It is, essentially, the ability of an agent to learn the irrelevance or repetitiveness of a specific stimulus over time and is characterized—it its simplest case—by a decreased (neuronal or local) response with each repeated presentation of the stimulus [77]. In terms of our robotic attention system, the activation of a location that has been habituated will be smaller than a location with novel, non-habituated data. Habituation effects have been observed in many animals with nervous systems and humans [78, 79, 80].

Although habituation in its simplest form is easy to define, there have been many observed variations. Several characteristics have been observed in studies [81] and are documented in [77] and [82]; some are non-associative —modulating a signal without any dependence on other signals or the overall context—and others are associative —where the expected signals are learned with respect to the current situation. In the latter case, the context can modulate the response either by inhibiting it directly (as in the non-associative case) or through the creation of expectations which are then compared to the actual environment. Below is a list of habituation characteristics and mechanisms (adapted from [77, 81, 82], and [83]).

- **Short-term Habituation**: sensory or motor neurons have a decreasing response when they are repeatedly activated.
- **Spontaneous Recovery**: habituation effects disappear when a stimulus has not been observed for some time.
- **Rate Sensitivity**: the recovery rate is faster when the stimuli occur repeatedly on a short time interval than when that interval is longer.
- **Savings**: when series of stimuli presentations are repeatedly followed by spontaneous recovery, the habituation rate increases within each series.
- **"Subzero" Habituation**: When stimuli is presented after habituation has reached its maximum value, the time until spontaneous recovery occurs is delayed.
- **Dimension Change**: Habituation effects disappear when a dimension of a stimulus changes—for example, the pitch or volume of an auditory signal.

- **Frequency and Intensity Effects**: Habituation rate increases when the stimulation frequency increases and the rate decreases when the stimulation intensity increases.
- **Stimulus Generalization**: Habituation to a specific stimulus leads to habituation to similar stimuli.
- **Dishabituation**: Habituation effects disappear when a novel stimulus is shown (at another location). Similarly, habituation effects also disappear when the context (the situation) changes.
- **Habituation of Dishabituation**: The dishabituation rate decreases when the stimulus that causes dishabituation is repeatedly presented.

A few computational models of habituation have been proposed. One oft cited model is Stanley's [84], which is described by a first-order differential equation (2), where y is the habituation over time, y_0 is the original value, τ is the rate of habituation, α is the recovery rate, and $S(t)$ is the stimulus.

$$\tau \frac{dy(t)}{dt} = \alpha(y_0 - y(t)) - S(t) \tag{2}$$

This model was used in or inspired many systems [85, 86] and was the basis of another model by Wang and Arbib [87]. They modified Stanley's model to include long-term habituation effects. Theirs is a coupled system of two equations, (3) and (4).

$$\tau \frac{dy(t)}{dt} = \alpha z(y_0 - y(t)) - \beta y(t)S(t) \tag{3}$$

$$\frac{dz(t)}{dt} = \gamma z(t)(z(t) - 1)S(t) \tag{4}$$

The first equation is similar to equation (2) with the addition of an input modulated by activity and a new gain, β. The second equation modulates the recovery rate with respect to the number of stimuli observed. The value of z is large after a small number of stimuli, which makes the recovery period fast; z becomes smaller as the number of stimuli increases, thereby increasing the recovery period. This causes the habituation to have longer-lasting effects.

The associative properties of habituation led Solokov [88] to develop a comparator theory of habituation, where stimuli are compared to an internal representation of the stimuli to determine habituation levels. A simple comparator model that exhibits rate sensitivity as well as a habituation rate increase with a stimulation frequency increase is presented in [83].

Balkenius [77] suggests that habituation must be a more general process since the internal representations are not always merely templates but sometimes involve complex cognitive processing based on the current context. His attention model incorporates habituation of irrelevant stimulus based on the context.

Novelty detection is important for robots, especially mobile ones. Habituation can be thought of as a type of novelty filter, essentially gating the activation of previously-observed stimuli with respect to new stimuli. Several applications therefore use habituation to detect novel features in the environment [89, 86, 90]. Others use habituation as an attention subsystem to create FOA shifts [33, 91, 77, 92], to classify spatio-temporal patterns [93], or even to improve navigation [94].

Many of the applications listed above implement habituation with a neural network [95] framework. Marsland et al. [86] use a Kohonen self-organizing map to implement a novelty filter that learns a model of the environment. Each neuron in the map is connected to a single output neuron with a habituable synapse that is governed by the model given in equation 2: stimuli that match the learned model receive a weaker synapse value than novel stimuli. Crook and Hayes [89] use a Hopfield network [96] to learn features; novel patterns have a higher energy than familiar ones.

Chang [94] added a component to a neural network algorithm to habituate to constant sensory data (such as a wall) to remove oscillations in the robot's movements when navigating narrow hallways. Sirois [91] uses a biologically-plausible neural network habituation model called HAB to replicate infant habituation functions on a robotic platform. Hebbian learning was used to orient the robot—through motor control—toward the most activated stimulus. Stiles and Ghosh [93] use a Habituated Multi-Layered Perceptron (HMLP) neural network to learn sonar patterns with extents in both space and time. It was shown that this system performed better than a Time Delay Neural Network (TDNN) because of its ability to encode long-term data.

All of the implementations described above must first be trained before they can be used. Several habituation systems have been implemented using techniques other than neural networks. For example, Breazeal and Scassellati [33] implement habituation for the robot Kismet as a feature map which initially increases the activation of the FOA and then decreases it (or habituates) until a new FOA is selected. Peters and Sowmya [90] have developed a "Surprise Function" for their people-tracking visual robotic system WRAITH. This function habituates to repetitive events by keeping a memory of the past brightness values of each pixels; a moving average is then calculated from these values and compared to the current pixel brightness. This absolute difference represents the *surprise* value, or the amount that the current pixel differs from its expected value, and is used to direct attention to new or (unexpectedly) changing areas of the scene. This system is said to habituate to repetitive motion, although no results are presented.

The habituation mechanism developed by Déniz et al. [92] uses auxiliary signals—in addition to Stanley's model—so that periodic signals can be habituated. This is the only system we have found so far that incorporates habituation to periodic events (and presents experiments and results). The auxiliary signals are obtained from the spectrogram of the input stimuli. The first-level auxiliary signal is a thresholded norm of a particular frequency's variance over time and can identify prolonged stimuli as well as stimuli changing with a fixed frequency. A second-level auxiliary signal is also calculated by identifying fixed-frequency changes in the first-level auxiliary signal, which indicate an input stimuli with repeated frequency modulations, such as a siren. This system was tested on audio signals and separately on image sequences, although other sensory modalities could be used.

Previous work by Hambuchen [10] on the SES attention network incorporated a method for the habituation of periodically repeating events. Sensory processing modules (SPMs) process information to detect occurring events; this processed information is then sent to a pre-filter that determines the salience of this particular event. The pre-filter then sends the event and associated salience to the SES to be registered. Salience is a combination of values representing incidence (salience that signals that an event has occurred at a particular location), task-relevance (increased salience for event that are related to the current task), and habituation (decreased salience for events that occur repeatedly at a particular location over a period of time determined by the developer).

The habituation value of a specific event v at a given time t is shown in equation (5).

$$H(v,t) = e^{-\beta_H S_t} \tag{5}$$

β_H is the habituation rate whose best value was experimentally determined to be 1, S_t is the time step for the event and is incremented for each occurrence of the event within the specified time period. This value is reset to 0 if the event does not reoccur during the time period. It is mentioned that a habitual event that does not reoccur should receive more salience as this may require attention; however, it is not clear that the location's salience is increased beyond its original non-habituated incidence and task-relevance values. Although co-occurring sensory events can be habituated, habituation is applied per event detected by a single SPM, and not to the overall event detected by multiple SPMs.

In her future work section, Hambuchen mentioned that the habituation value should only modulate the incidence and not the task-relevance portion of the salience so that habitual events related to the task can be attended. Moreover, habituation should not be applied to an event upon its first few occurrences; instead, the event should repeat a specified number of times before habituation is applied. She also suggests that the salience of co-occurring events should be increased instead of decreased, so that situations such as a robot continuously running into a wall—which would be detected by more than one sensory mode—can be attended and remedied.

In summary, the systems presented above do not combine multiple sensory modalities to guide habituation. The systems perform habituation at the sensor level; however, when multiple modalities are combined, it may be necessary to habituate at a higher level, where task-relevance influences the process. This is explored in our work. Only two of the systems mentioned above can detect periodic events; this is an important part of our system, as is the ability to determine what time interval makes a particular event habitual. Habituation is intrinsic to our attention system. It is not a novelty detector, but instead suppresses the activation of nodes by repetitive and irrelevant stimuli so that the sensitivity of the FOA to those events is diminished and the robot can avoid unnecessary FOA shifts. Notably, our system exhibits some of the characteristics of habituation enumerated above.

CHAPTER III

SYSTEM DESCRIPTION

This chapter describes the logical and computational structure of our attention system. A general overview of the system is given first, followed by more detailed discussions of its local and global processing streams. Lastly, the habituation component is discussed.

Attention System Overview

The software module at the heart of the attention system is the Sensory Ego-Sphere (SES), an egocentric memory structure for sensory information (cf. Chapter II). The attention system comprises several interacting modules that could serve as inputs to higher-level processing modules, as shown in figure 4. The modules inside the large square represent the implemented system.

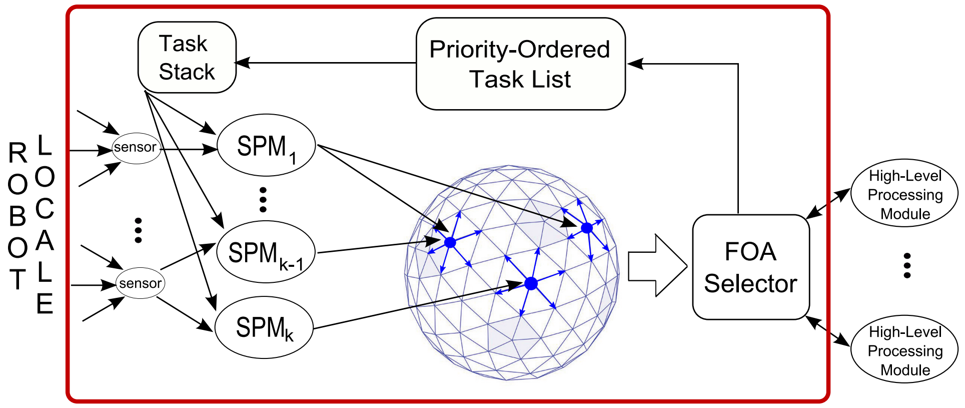

Task Stack
Priority-Ordered Task List
ROBOT
LOCALE
sensor
sensor
SPM$_1$
SPM$_{k-1}$
SPM$_k$
FOA Selector
High-Level Processing Module
High-Level Processing Module

Figure 4: Diagram of the attention system

Specifically, the attention system involves the following modules: The first is the SES itself, which is made up of nodes with (potentially) individual computing power and communication capabilities with their neighbor nodes. There are also various sensory processing modules—abbreviated as SPMs—that write data to nodes on the SES. The FOA selector module—abbreviated FOA—does so as a function of the information posted on the SES by the SPMs. Then there is a task module that contains a list of the features required by the active task. These features are given in terms of the sensors/SPMs used in the system.[3] The task module is implemented as a stack so that higher-priority tasks can interrupt the current task when necessary.

Figure 4 shows the information flow: An event occurs in the robot's locale and is detected by those sensors that are sensitive to the stimuli. The sensors send these raw data to various SPMs for abstraction. Sensor types are determined by the purpose of the robot, its environment, the tasks it is to perform, and the dangers it may encounter. Examples include vision, motion, face detection, and sound localization. The SPMs are parameterized for the current task. They calculate an activation value based on their inputs and the task parameters and post the value to the appropriate SES node(s) along with a time stamp and the acquired data itself. Based on the resolution of the SPM, the registration node (the node that has received data directly from the SPM) may spread the data and activation to adjacent nodes. The FOA selector then uses the overall activation values of each SES node to determine the most salient node. In the absence of any large changes of activation elsewhere on the network, this node becomes the focus of attention. The node's data are then compared to a list of higher-priority tasks; if they matches one of these tasks, this task is pushed onto a task stack and becomes active. Although not implemented in this system, the FOA information could be passed on to other modules for higher-level processing such as object recognition.

Since habituation is intrinsic to the system's nodes, there is no global habituation module shown in the system diagram. Habituation variables are defined for each sensor type at each node since sensors can have different resolutions and update

[3]For example, if the SPMs are a color segmentation module, a motion detector, and a face detector, the task's features would be defined as the color of the target, whether it moves or not, and whether or not we can expect a face to be associated with it.

rates. These variables can be modified by the FOA selector to bias the behavior toward sensitization or habituation; they can also be reset by the task module upon a task change. This is described in more detail in the habituation section below.

SES Implementation

The sensory data gathered in this work was obtained using the robot ISAC's sensors. ISAC is a humanoid robot developed at Vanderbilt University, and is equipped with pan-tilt units fitted with color cameras. Color, motion, and face recognition SPMs use 320x240 pixel images grabbed from the cameras as their raw sensory input. They post processed data and activation to the SES. The SES is centered around ISAC's cameras and the position of the pan/tilt units is used to determine the registration node.

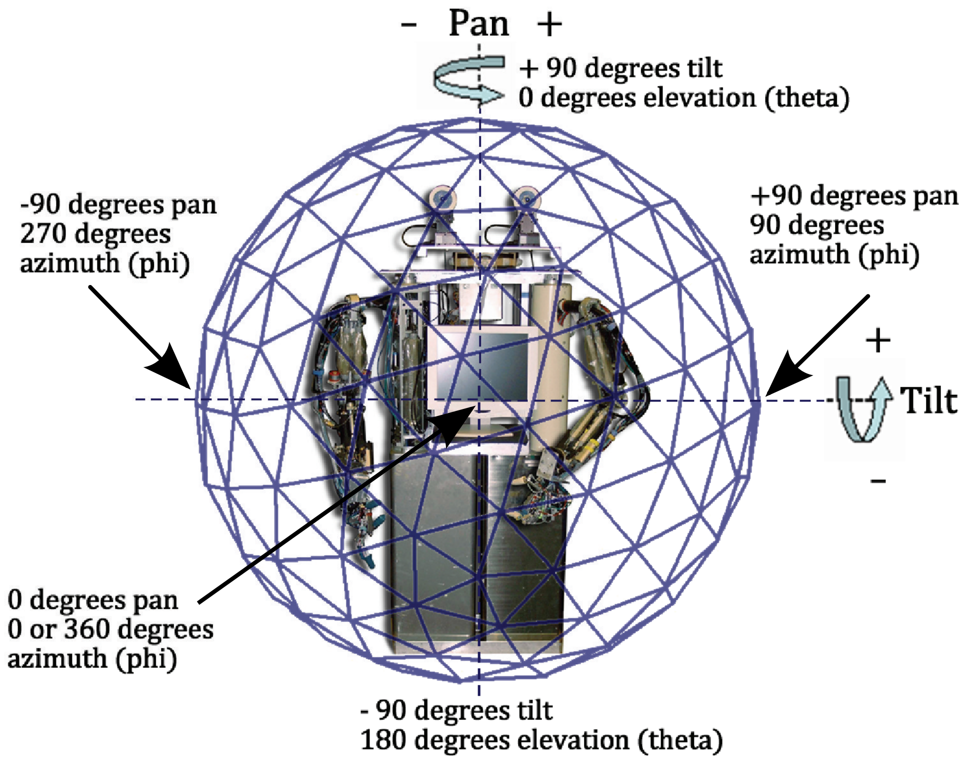


Figure 5: ISAC in its Sensory Ego-Sphere

Figure 5 illustrates ISAC within its SES as well as the conversions between pan/tilt angles and azimuth/elevation angles that must be performed to post data to the appropriate node.

As can be observed from this figure, θ ranges from $0°$ at the North pole to $180°$ at the south pole. ϕ ranges from $0°$ at the front center of the SES to $360°$ at the same point, moving in a counter clockwise direction around the vertical axis.

The SES used in this work has a tessellation of 14, which yields 1962 nodes where information can be stored. The nodes are spaced between 4 and 6 degrees apart, depending on their location on the sphere.

Although the SES is currently a serial implementation, we would like to explore the possibilities of implementing it in parallel, such that each node is a simple processor that communicates only with its immediate neighbors. For this purpose, hexagonally-connected networks such as the SES have some advantages over rectangular ones. Every node in the interior of a hex-net has 6 equidistant neighbors. Let l represent that distance. In a rectangular net each node has either 4 neighbors at a distance l (in a 4-connected topology) or 4 at l and 4 at $\sqrt{2}l$ in a 8-connected topology. If the network is to be implemented in hardware, the signal latency between nodes is proportional to the distance between them. Having equidistant neighbors therefore simplifies the timing of the devices. A hex-net is symmetric with respect to $n \times 60°$ rotation, whereas both types of rectangular net are symmetric with respect to $n \times 90°$ rotation. The relative advantages of hexagonal connectivity over rectangular for the purpose of parallel computation have been known for some time. (See, for example, [97] or [98].)

Local and Global Activity

There are two levels of activity in this attention system: the local interactions on a global scale resulting from the interconnection of the nodes on the SES make up the bottom level, and the global interactions of the FOA selector module to direct attention constitute the top level. Figure 6 gives an overview of the local and global calculations taking place in the system. As can be seen from this figure, the FOA selector, task list, and task stack all operate globally over the entire system. At the local level, SPMs post data to individual nodes and nodes pass information on to their immediate neighbors through a process called *spreading activation procedure*. The nodes asynchronously receive information from the SPMs when that information is made available and update their activation values. On the other hand, the FOA module selects a new focus of attention after a predetermined time interval.

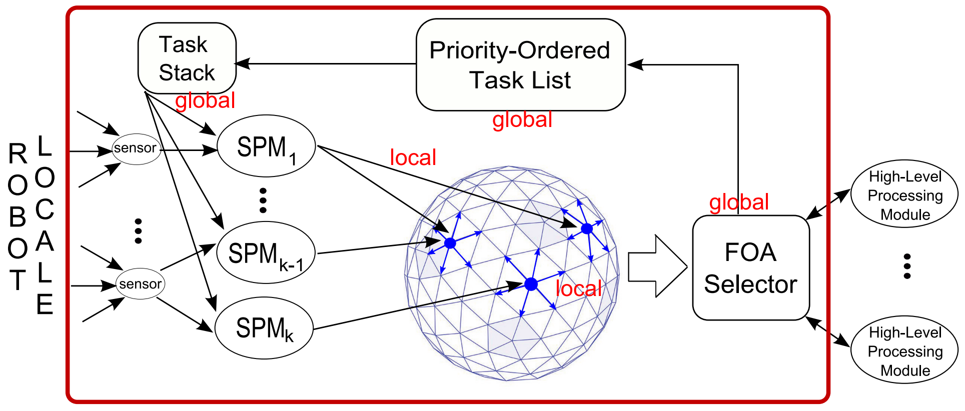


Figure 6: Local and global activity in the system

The following two sections contain more in-depth details about the system's modules based on local and global activity levels.

Local Activity

Several calculations take place at the node level each time new information is posted on the SES. To facilitate these calculations, there are various variables defined at each node of the SES. Figure 7 lists them.

Each SPM has its own set of variables defined at each node. First, there are two record structures: a *currentTag* and a *previousTag*. When new sensory information arrives at the node—whether this posting comes from a SPM or a neighboring

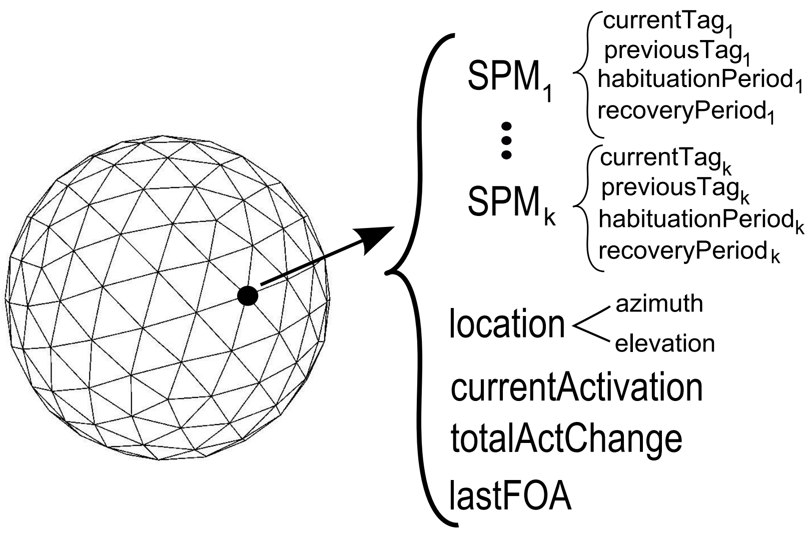


Figure 7: Variables defined at each node

node—the information is stored in the record structure. Table 2 lists the record's components, which consist of the data's location, the activation calculated by the SPM, a timestamp of the data as well as the raw data itself. Note that the actual location of the data itself is not lost: although the SES partitions the locale into distinct regions centered around a node, the data's location is retained and used in calculations such as spreading activation and habituation.

Table 2: Information stored in each SES tag (at each node and for each sensory modality)

Name	Description
location	pan/tilt location of the data w.r.t. the robot
timestamp	time at which the data was recorded by the SPM
activation (*act*)	measure of how closely the data matches the current task, calculate by the SPM
change in activation (Δ_{act})	measure of change at the node based on the current and previous data
absolute activation	activation measure not dependent on the current task
data	data received and (potentially) pre-processed by the SPM

Additionally, each SPM also has its own *habituationPeriod* and *recoveryPeriod* variables defined at each node. These variables dictate the duration of the habituation effects and will be explained in further detail in the habituation section of this chapter.

The non-SPM related variables include a *location* variable that holds information about the location of the node on the SES, expressed in azimuth and elevation coordinates.[4] The *currentActivation* and *totalActChange* variables hold values representing the sum of the task-based activation and absolute change in activation associated with all data posted at the node by the SPMs; these values are used by the FOA selector module to determine the new focus of attention. Lastly, the *lastFOA* variable is used to keep track of the last 2 timestamps of the node's selection as focus of attention. These are also used in the habituation procedure.

As mentioned previously, the SES implementation used in this work contains nodes with a 2-cell memory for each SPM. Each node holds the current record of information as well as one previous record of information. When new information arrives at the node from a SPM, the information in the node's *currentTag* for that SPM must be moved to the corresponding *previousTag* before the new information can be posted.[5] Activation is then spread to the node's neighbors based on the SPM's resolution; this procedure is detailed later in the section.

The next step consists of adjusting the activation of the new information to incorporate habituation. Although SPMs do calculate an activation value for the information that they post to a node, it is necessary to adjust this value to generate the desired habituation effects. The information stored in the current and previous tags, along with the habituation variable defined for each node, are used for these calculations. Habituation is only incorporated if the new information is identical[6] to the previously-posted information found in the *previousTag*. The habituation effect

[4]See the section on the SES in chapter II.

[5]The last and before-last postings are kept in memory regardless of posting time. When a new event occurs, the information stored in the *currentTag* is moved to the *previousTag* and what was in the *previousTag* is discarded. For example, if SPM k posts information at node n, then $currentTag\,\{n,k\} \rightarrow previousTag\,\{n,k\}$ and $newInfo\,\{n,k\} \rightarrow currentTag\,\{n,k\}$.

[6]Consecutive postings on the SES at a particular node can theoretically be "identical" . In practice however, there exists a certain amount of sensor noise ϵ which renders absolutely identical

is based on the node's habituation variable for the corresponding SPM as well as the time difference between the current and previous posts. The exact procedure is detailed in the habituation section below.

Once habituation has been applied to the activation value, the activation and change in activation for that SPM's tag are updated. Upon a change in these values for any SPM, the total activation and total change in activation for the node are recalculated and made available to the FOA selector module.

Pseudo-code for all calculations taking place at the node level is shown in figure 8 below.

```
Upon posting at node $n$ from SPM $k$:
$currentTag\{n,k\} \rightarrow previousTag\{n,k\}$
$newInfo\{n,k\} \rightarrow currentTag\{n,k\}$
$SpreadActivation(n,k)$ // function detailed below
if $currentTag\{n,k\}.data == previousTag\{n,k\}.data$ then
    $CalculateHabituatedActivation(n,k)$ // function detailed in
        habituation section
    Update total activation
    Update total change in activation
end
```

Figure 8: Node Posting Procedure

Spreading activation procedure

Sensory data gets posted by a SPM to a single node on the SES. However, it is a possibility that the resolution of the data and the resolution of the SES do not match. Specifically, low-resolution sensors such as microphones have localization errors larger than the SES resolution itself. If that is the case, the registration node may not be the true node that should be associated with such data; there could be a neighboring

postings improbable. What is then meant by "identical" or "the same" throughout this work is in fact "identical within an error margin ϵ."

node whose location is closer to the actual data. Therefore, it is important for the activation caused by such sensors and registered by their corresponding SPMs to be spread to neighboring nodes until the sensor's resolution is reached. This is necessary so that activation computed by these SPMs can correctly combine with other SPM activation to guide the FOA to the appropriate node.

A spreading activation procedure for the SES was developed in [10]. In that procedure, decayed activation was spread to all nodes within the area of the posting SPM's resolution serially. We have adapted this procedure here to implement a distributed spread in which each node passes on activation to a receiving neighbor node based on the direction in which it received its activation. A distributed scheme will be useful in the future if the SES takes on a parallel implementation with each node having its own computing power.

We are currently using a simple scheme to spread the activation from the registration node to the neighboring nodes. To make use of the SES's interconnections, the activation at a registration node is decayed and spread to the node's direct neighbors. These first-level neighbors then spread a further-decayed activation value outward in the same direction as its own received activation. An example is shown in figure 9 where activation is spread from the blue node to the green node.

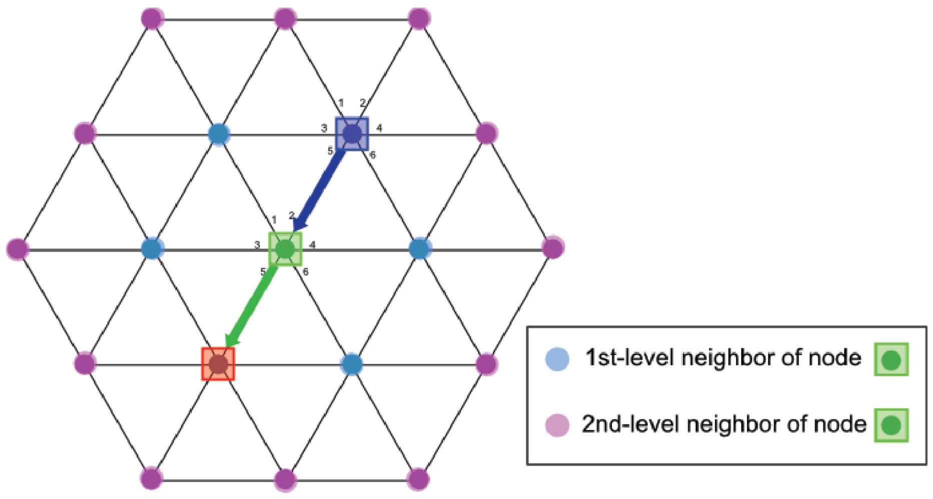


Figure 9: Node Communication.

A consistent neighbor numbering scheme is used in this figure and in our work. From the green node's point of view, activation has been received from neighbor number 2; the green node then spreads the activation to its neighbor number 5, or the red node in the figure, which will be receiving from its 2nd neighbor. As the activation is spread farther away from the originating node, each new level has an increasing number of neighbors: a node has 6 direct first-level neighbors (as indicated by the light blue nodes on figure 9), at the second level there are 12 neighbors (the magenta nodes on figure 9), and so on. Because of this node expansion, certain nodes will spread activation to two different neighbors. One is a direct neighbor and can be linked back to the originating node through one of its first-level neighbors, as shown by the green line of communication in figure 10.

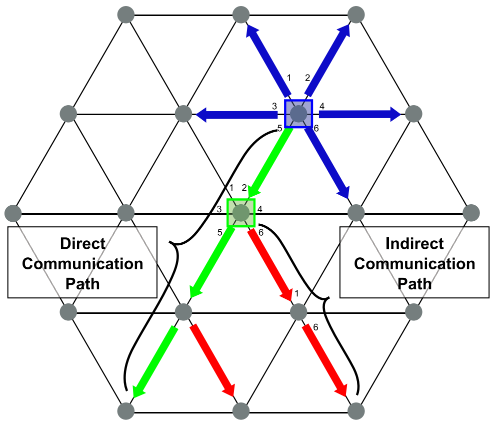


Figure 10: Direct and Indirect Activation Spread.

The other type is an indirect neighbor and branches off the direct line of communication, as shown by the red line. Each indirect neighbor will only spread activation to one neighbor following the scheme presented in figure 9. Direct neighbors outside of the first-level will spread activation to two neighbors: one direct neighbor following the scheme of figure 9 and an indirect neighbor determined by the direct neighbor selection. The activation spread is repeated until the decayed activation reaches a value less than one. The rate of decay is dependent on the sensor type that made the original posting: high-resolution sensors have a fast decay rate and, therefore, limited to no spread. Low resolution sensors have a greater localization error and must be spread to more nodes to adequately cover the area from which the data could have originated.

The equation for calculating the exponentially decaying activation Act_d is shown in equation (6) and takes into account the distance between the current node to which activation is being spread and the original registration node ($Dist$), as well as the sensor inaccuracy or resolution. The value of α is calculated for each sensor so that only 1% of the activation remains at the limit of the sensor's resolution range, as shown in equation (7).[7] This is the termination condition for the spread procedure.

$$Act_d = Act_0 \ e^{-\alpha Dist} \tag{6}$$

$$\alpha = \frac{-ln(0.01)}{Resolution} \tag{7}$$

When activation is spread to a node, the original data is also spread. This is done so that modified activation values can be calculated based on the habituation constants at the node.

Global Activity

A diagram of the FOA selector module in terms of its software components is shown in figure 11. The *activation table* is a repository of information about the total activation (across all sensory modes) and change in activation at each node on the SES. It is updated each time the information at a node changes. The FOA selector polls this table to find the node location with the maximum activation as well as the location

[7]Since the activation is exponentially decaying, it will never reach 0. Therefore, 1% of the original activation is picked as the termination condition and the α constant for that sensor is computed with that in mind

with the biggest change in activation: these values are then used to determine whether the current focus of attention is maintained or whether it is shifted to a new location. Once a new FOA location is selected, it is sent to the priority-ordered task list module for comparison. If the location's features matches those of a higher-priority task, that task will then be sent to the task stack module to become the current task. If the location matches neither the current task nor any higher-priority tasks, it will be habituated by the system. Details on this procedure are found below.

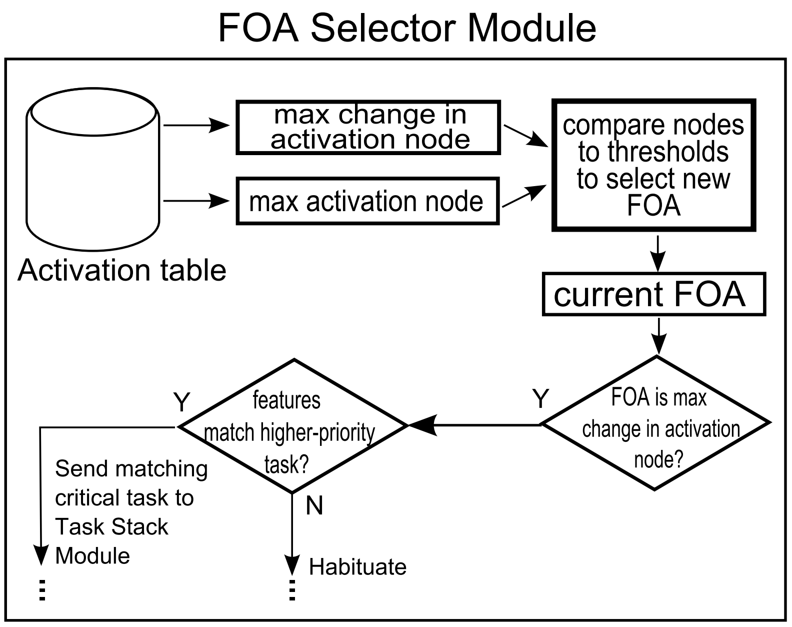

FOA Selector Module
Activation table
max change in activation node
max activation node
compare nodes to thresholds to select new FOA
current FOA
FOA is max change in activation node?
Y
features match higher-priority task?
Y
N
Send matching critical task to Task Stack Module
Habituate

Figure 11: FOA selector module

The FOA selector module makes the decision either to remain focused on the current location or to shift the attention focus to a new location; this decision is made at each clock cycle. Two criteria guide the FOA selection: maximum activation and maximum change in activation. The maximum activation ($maxAct$) gives the location of the most salient location on the SES based on the current task; the maximum change in activation ($maxChange$) gives the location that has undergone the largest absolute change in activation, and represents a potential need for a task switch. Both the maximum activation and the maximum change in activation are obtained from the *activation table*.

Depending on the actual values of the *maxAct* and *maxChange* variables, the FOA could either shift to a new node or maintain its current position. Additionally, a higher-priority task could either be pushed on the task stack to become the current task or be removed from the stack, making the initial task active again. It is also a possibility that the current task will remain unchanged.

Along with the *maxAct* and *maxChange* variables, three thresholds are involved in this selection process; they are *actChangeThresh*, *criticalThresh*, and *maxThresh* and are used primarily to facilitate the task-switching behavior of the system. Their definitions are as follows:

- **criticalThresh**: threshold below which the maximally activated node (*maxAct*) must be to deem the situation "resolved" and pop the critical task off the task stack and return to the initial task.

- **actChangeThresh**: threshold above which the maximum activation change node (*maxChange*) must be to send the node's data to the priority-ordered task list module for comparison. This indicates a potential task change. If *maxChange* is below this threshold, then the node's data is not compared to higher priority tasks since there is not change indicating new data in the locale.

- **maxThresh**: this threshold is also used to determine whether a node's data is sent to the priority-ordered task list module. When a FOA is selected by the *maxChange* criteria, that node's current activation is compared to the *maxThresh* threshold; if the value is above this threshold, the shift is attributed to the current task and not to a potentially new task since the node's features strongly match the current task's description (so data is not compared to critical tasks and the current task remains active).

The FOA selection procedure is as follows:

1. Determine whether a task must be popped from the task stack:
    (a) If the current task is a task that has been loaded from the priority-ordered task list and not the initial task, compare *maxAct* to *criticalThresh*. If the maximally activated node is below this threshold—meaning that no location on the SES closely match the higher-priority task—we can assume that the situation matching the higher-priority task has resolved and that it is now safe to go back to the initial task.
2. Determine what the new FOA is:
    (a) Compare *maxChange* to *actChangeThresh*. If the node with the maximum change in activation is higher than this threshold, then the FOA shifts to this node (or stays at this node if it was there already).
    (b) If *maxChange* was below the *actChangeThresh* threshold, FOA shifts to the *maxAct* node (or remains there if it was there already).
3. Determine what task the FOA shift is attributed to:
    (a) If *maxChange* was greater than *actChangeThresh*, compare the *maxChange* node's total activation to *maxThresh*. If it is above threshold, we attribute the FOA shift to the current task and the node's data are not sent to the priority-ordered task list for comparison. If it is below threshold, we attribute the FOA shift to a potentially higher-priority task—because the node did not closely match the current task—and the node's data are sent to the priority-ordered task list module for comparison.

(b) If *maxChange* was less than *actChangeThresh*, the shift is attributed to the current task and the node's data are not sent to the priority-ordered task list module.

This procedure is expressed in pseudocode below.

```
// Check if higher-priority situation has resolved
if $higherPriorityTask == active \ \&\& \ maxActNode_{act} < criticalThresh$
then
    // Situation has resolved
    $Remove\ higher-priority\ task\ and\ return\ to\ initial\ task$
end
else if $higherPriorityTask == active \ \&\& \ maxActNode_{act} \geq$
$criticalThresh$ then
    // Situation has not resolved
    $Keep\ higher-priority\ task\ active$
end
// Select New FOA
if $maxChangeNode_{\Delta_{act}} > actChangeThresh$ then
    // Maximum change in activation node is the new FOA
    $NewFOA = maxChangeNode$
    // Determine whether we need to compare to higher-priority
        tasks
    if $maxChangeNode.act < maxThresh$ then
        $match$ = CompareNodeToHigherPriorityTasks()
        if $match$ then
            $Put\ higher\ priority\ task\ on\ stack\ and\ make\ active$
        end
        else if $!match$ then
            $Keep\ the\ current\ task\ active$
        end
    end
end
else
    // Maximum activation node is the new FOA
    $NewFOA = maxActNode$
end
```

Figure 12: Focus of Attention Selection and Task-Switching Procedure

If the new FOA is attributed to a potentially higher-priority task, the node's data is sent to the priority-ordered task list module. There, it is compared to a list of tasks that have a higher-priority than the current task. If the node matches one of these tasks, the task is pushed onto the task stack and its features are then compared to incoming data at the SPM level to determine activation levels on the SES. However, if the node does not match any of these higher-priority tasks, the FOA shift is deemed a false alarm and the node is considered a distractor node and is habituated. To detect situations matching tasks with higher priority that the selected initial task at the beginning of an experiment, all tasks having a priority higher than the initial task are pushed onto the task stack (in order of priority so that the highest priority task is active first). If information in the locale matches these higher-priority tasks, they will remain active; if, however, no information matches these tasks, they will be removed from the stack and the initially selected task will become active. This is done to prevent the system from failing to detect high-priority situations that existed before initialization and that may not be detected by the change in activation variables (since they may not be changing).

When a node is selected as FOA, the time is saved in the node's *lastFOA* variable. This will be used by the habituation procedure described in the next section.

Habituation

There is no dedicated habituation module; instead, habituation is incorporated throughout the overall system. Habituation operates at the local level—at each node—and at the global level—through the FOA selector.

Habituation at a node depends on several factors. First of all, only data reoccurring at a node is habituated. If a node receives data that is different than what it previously received, then there is no habituation. This is done because we wish to habituate to the actual stimuli itself, and we want to be able to detect changes in the locale since these could indicate an important or dangerous situation that needs attending to. Secondly, the amount of habituation at a node depends on the time that has elapsed between the new posting and the previous posting; habituation can range from total habituation to no habituation at all. Lastly, whether or not a node has been selected as FOA in the past affects habituation. If a node has never been selected as FOA, it is deemed not important enough to become a distractor even at its full value and, therefore, it is not habituated. This is done to save computing time. If however, a node has been the focus of attention in the past, its stimuli may be strong enough to shift the FOA onto it once again, and it must be habituated. The specific way in which these factors influence the habituation procedure is detailed below.

When new sensory information is posted to a node, the current and previous tags are compared to determine whether they correspond to the same stimuli. Habituation comes into play when these two sensory events are the same. Habituation is calculated using two variables defined for each SPM at each SES node (shown in figure 7). These variables modulate the activation change at the local level. The *habituationPeriod* variable (abbreviated t_{hab}) sets the period of time that a node is habituated by keeping its change in activation low (in that sensory modality). The *recoveryPeriod* variable (abbreviated t_{rec}) sets the length of the time period during which the node's change in activation returns to its full value.[8] These variables are used to define three stages of operation. The change in activation (Δ_{act}) variable of each *currentTag* is modified based on the stage in which the posting occurred. The stages and variable modifications incurred are explained below. Selecting the correct stage involves computing the timestamp difference (abbreviated TSD) between the data in *currentTag* and *previousTag*. The modulation of the change in activation variable is the most important part of the procedure since it is this variable that could repeatedly attract the FOA although the location may not match the task. We are

[8]In our implementation, the *habituationPeriod* variable and the *recoveryPeriod* variable always have equal length and are initialized to a value smaller than the time between two consecutive data postings.

not as concerned with the modulation of the activation variable, since it is dependent on the current task.

- Habituation Stage: The time period where the node is completely habituated. This occurs when the timestamp difference between the current posting and the previous posting is less than the *habituationPeriod* variable ($TSD \leq t_{hab}$).
  - Change in Activation is modified: $\Delta_{act_k} = 0$
- Recovery Stage: The time period where the habituation effects begin decaying and, therefore, the node's change in activation begins returning to its full value. This occurs when the timestamp difference between current and previous postings is greater than the *habituationPeriod* but less than the sum of the *habituationPeriod* and *recoveryPeriod* ($t_{hab} < TSD \leq t_{hab} + t_{rec}$).
  - Change in Activation is modified:
    $\Delta_{act_k} = |Act_k(t_{current}) - Act_k(t_{previous})| * Decay$
    where $Decay = \frac{(TSD - t_{hab})}{t_{rec}}$
- Normal Stage: The time period where the node is completely free of habituation effects. This occurs when the timestamp difference between current and previous postings is greater than the sum of the *habituationPeriod* and *recoveryPeriod* ($TSD > t_{hab} + t_{rec}$).
  - Change in Activation variable is not modified

The decay modulating Δ_{act_k} in the recovery stage is linear. If data comes in at the beginning of this stage, Δ_{act_k} will be close to fully habituated; if data comes in near the end of this period, Δ_{act_k} will be close to its original value. Figure 13 illustrates the habituation effects on the change in activation variable through all three stages.

Note that this procedure is only performed when a node has already been selected as FOA in the past. This is so that time is not wasted on nodes that are not activated

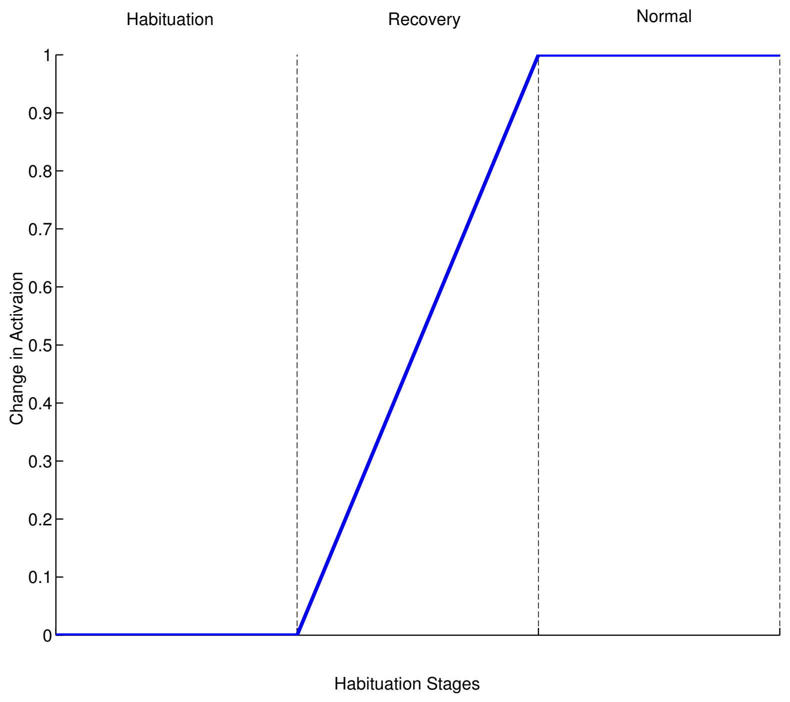


Figure 13: Effects of Habituation Stages on Change in Activation

strongly enough to ever become the focus of attention. Once a node becomes the FOA, a timestamp is recorded in the node's *lastFOA* variable, shown in figure 7—the last two timestamps are kept at the node much like *currentTag* and *previousTag*. These timestamps are used to indicate that a node has been FOA in the past and should be habituated if need be.

While the above procedure operates at the local level, the values of the *habituationPeriod* and *recoveryPeriod* variables can be modified at the global level by the FOA selector. These values are initialized to be very small values (smaller than the time between two consecutive postings so that there is no habituation at first). Based on certain situations occurring in the system, the FOA selector can modify these variables or reset them to their initial values to derive the appropriate behavior. These situations are summarized below.

- Habituation variables for all nodes are reset when there is a task change in the system:
    - A higher-priority task is pushed onto the task stack for more than one update cycle.
    - A higher-priority task is popped from the task stack and the initial task becomes active again.
- Habituation variables are modified for FOA node when:
    - A higher-priority task was pushed onto the task stack for one update cycle only due to the FOA selection
    - A FOA is selected because $maxChange$ was greater than $actChangeThresh$—indicating a potential task switch—but the data did not match any higher-priority tasks

As explained above, habituation variables are reset for all nodes on the SES when a task change occurs. When a FOA is selected by the maxChange criterion but it does not matching any higher-priority tasks or it causes a task to be pushed onto the stack and immediately popped off, that node's habituation variables are modified so that it will be habituated in the future. The habituation variables are modified as follows:

- If the node has been FOA at least once prior to now (not counting its current selection as FOA)
    - *habituationPeriod* (and *recoveryPeriod*) is set to the time difference between now and the previous FOA time.
- If the node has never been selected as FOA prior to now

- *habituationPeriod* (and *recoveryPeriod*) are incremented by a predefined time period, $addTime$, chosen to correspond to the time between consecutive data postings.[9]

Once a node's habituation variables for a specific SPM are modified, the variables' values are then spread to the node's neighbors (based on the sensor resolution) according to the spreading activation procedure detailed in the previous section. Although the activation is decayed as it is spread, the habituation constants are passed unchanged.[10]

Habituation variables are modified in one final manner. Since our goal is to detect changes in the environment, we would like to be able to detect a change in the periodicity of a distractor. This means noticing changes in the length of time between distractor sightings. An increasing period between consecutive distractor sightings can already be detected with the system as described above: the second distractor sighting will fall outside of the habituation period and will receive either a percentage of its activation if it falls within the recovery period or full activation if if falls within the normal operation period—the period having increased to more than double what it was before. However, a period that is decreasing will not be detected since the second distractor sighting will always fall within the habituation period and will be fully habituated. Because of this, the habituation variables are decayed as the time since a node was last FOA increases. The variables will therefore return to their initial values over time and a change in the period between consecutive distractor sightings will be detected as soon as the habituation variable's value becomes less than the period between distractor sightings.

[9]$addTime$ is equal to 0.05 seconds in our implementation, which corresponds to the minimum time between consecutive postings of the highest rate signal.

[10]Habituation constants are spread because the uncertainty that determines the spreading activation network is such that a situation could arise where a SPM posts information to the SES at a neighboring node m of the original registration node n. In that case, we want to be able to apply the habituation calculated at node n to node m.

Characteristics of Habituation

Four of the characteristics of habituation identified in chapter II are addressed in this system. The first is short-term habituation, defined simply by a decrease in response with repeated presentation of a stimulus. This is accomplished by modulating the activation change variable Δ_{act_k} in the habituation and recovery stages. When a distractor is repeatedly presented on a regular interval, the activation given to the node is decreased and the node will not be attended by the FOA.

The second characteristic of habituation is spontaneous recovery, which states that the effects of habituation disappear when a stimulus is not seen for a period of time. This is addressed by the normal stage of habituation which allows Δ_{act_k} to take on its full value once a distractor occurs after a time period greater than the habituation and recovery periods combined.

Another characteristic addressed by the system, subzero effects, delays the time until spontaneous recovery when a stimulus occurs after habituation is at its maximum value. This is implemented by restarting the habituation period each time a distractor is observed at a node; the more times it is observed, the longer it will take until Δ_{act_k} returns to its full value.

Lastly, the system implements dishabituation, defined by a disappearance of habituation upon a new stimulus presentation or context change. This occurs when there is a task switch in the system, either because a target matching a higher-priority task shifts the FOA or is removed from the locale. In these cases, the *habituationPeriod* variables are reset across the SES, thereby removing all habituation effects.

CHAPTER IV

EXPERIMENTS AND RESULTS

Parameter Selection

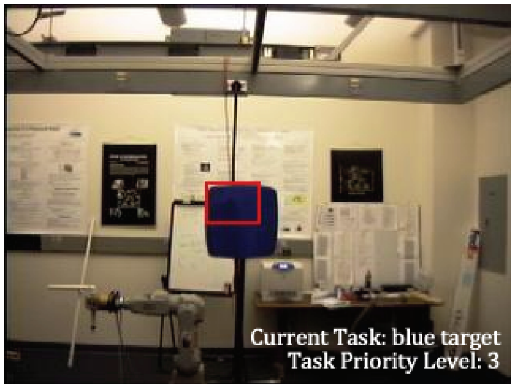
An experiment was performed to identify the best parameter values to use in the algorithm. The three parameters are thresholds used in the procedure that identifies the new focus of attention, as described below. Each parameter can take on a value between 0 and 1.

- **criticalThresh**: threshold below which the maximally activated node must be to deem the situation "resolved" and remove the higher-priority task from the task stack.
- **actChangeThresh**: if the node with the highest *actChange* value is above the *actChangeThresh* threshold, then there is the potential for a task change and the node's data is sent to the higher-priority task list module. If the highest *actChange* on the SES is below this threshold, then the node's data is not compared to higher-priority tasks.
- **maxThresh**: when a FOA is selected because of its large *actChange* value, its *currentAct* value is compared to the *maxThresh* threshold; if the value is above this threshold, the shift is attributed to the current task and not to a potentially new task (so data is not sent to the higher-priority task list module for comparison and the current task remains active).

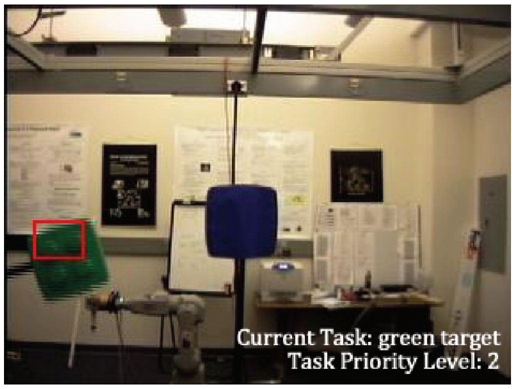
Because these parameters deal with potential task-switching situations, a real-world dataset was created that includes task-switches. The dataset was designed to be neither a trivial case nor the most difficult situation that could be encountered, but rather something in the middle. In the situation, there are three targets consisting of blue, red, and green squares. The blue square is the current task, which has a priority level of 3. The green square has priority level 2 and the red square has priority level 1 (the highest priority). At the start of the situation, the blue square is visible and the system should attend to it. A few seconds later, the green target enters the locale and the system shifts its current task to this target since it has higher priority. Similarly, the red target enters the locale and the system shifts to the red task since it now has the highest priority. The red and green targets exit the locale one at a time and the task returns to the blue target.

Figure 14 shows selected frames from the dataset as well as which task is/should be currently active. When only the blue target is present in the locale, the system should select the blue target as the current task and should attend to it. Once the green target enters the locale (figure 14b), the task should shift since the green target has a higher priority than the blue target. Similarly, once the red target enters the locale, the task should shift to attend to it since it has the highest priority in the situation (figure 14d). However, as shown in figure 14c, the system does not detect the red target in this frame and incorrectly attends to the green target instead. This is corrected in the next frame. Once the red target leaves the locale and the green and blue targets remain, the task should shift to attend to the green target (figure 14e). Similarly, once the green target leaves the locale leaving only the blue target, the task shifts to attend to the blue target (figure 14f).

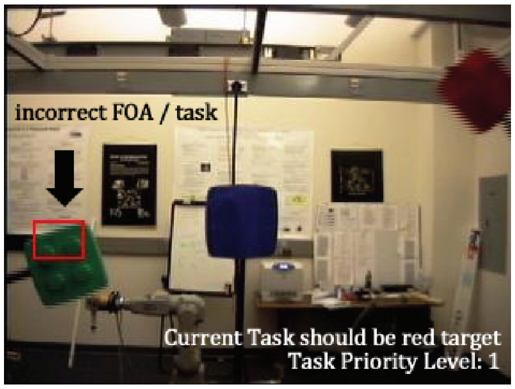
To test all possible parameter combinations, each parameter must be varied from 0 to 1 in 0.1 increments. A previous experiment narrowed this range to 0-0.6 for *maxThresh*, 0-0.4 for *actChangeThresh*, and 0-0.7 for *criticalThresh*, for a total of 280 unique parameter combinations. Therefore, 280 trials were performed, each with a different parameter combination. For each trial, the position of the FOA in each frame was recorded as well as the currently active task. This was then compared to the true node location of the highest priority target to get a measure of the system's



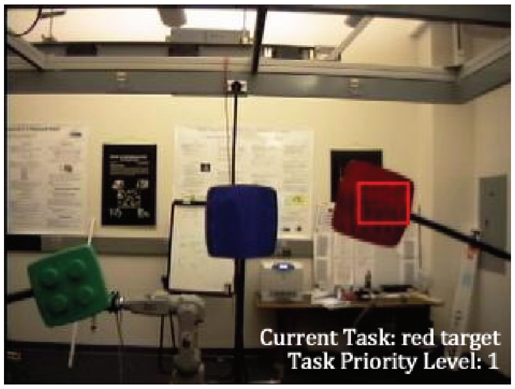
(a) frame 15: Blue target



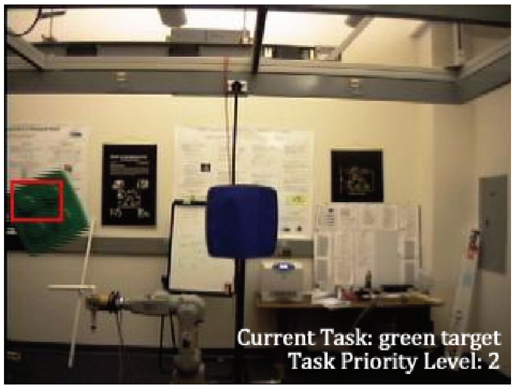
(b) frame 66: Blue and green targets



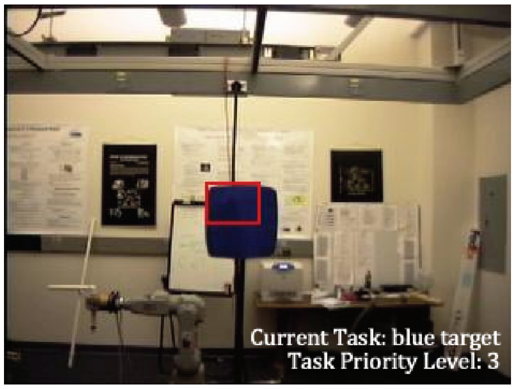
(c) frame 80: Target incorrectly selected



(d) frame 124: Red target correctly selected



(e) frame 201: Red target has left locale



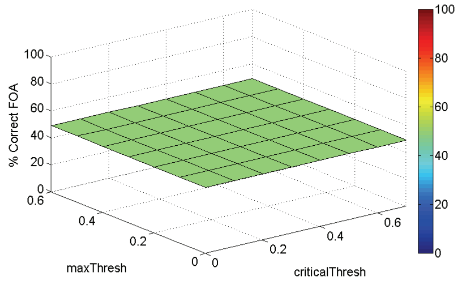
(f) frame 238: Green target has left locale

Figure 14: Frames from parameter selection simulation.

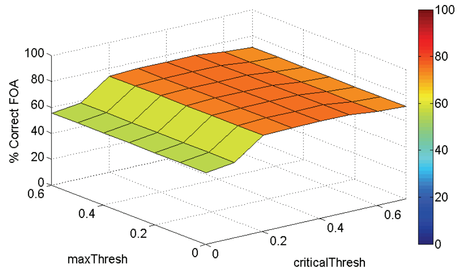
performance. Two different criteria were used: the *FOA criterion* evaluates the system by the percentage of correct FOA and the *task criterion* evaluates the system by the percentage of correct current tasks. A "correct" FOA is considered to be any node that overlaps on the target, and there can be more than one correct FOA per frame. The "correct" task is the task corresponding to the highest priority target visible in the locale. There is always a single correct task per frame. Statistics on "incorrect" FOA were also calculated to show the percentage of time that the FOA is on an incorrect target versus the background. The results are shown in figures 15 and 16. Because of the three-dimensional nature of the data, each contour graph shows the performance of the system—measured by the FOA criterion in figure 15 and by the task criterion in figure 16—at all values of the *maxThresh* and *criticalThresh* parameters for a given *actChangeThresh*. Orange areas correspond to a performance of 80% or more and red areas to 90% or more. Therefore, the redder an area of the graph is, the better the performance is for that parameter set.

As can be seen from the graphs, the parameter *maxThresh* (on the y-axis) has little to no effect on the performance of the system. This parameter was therefore fixed at a mean value of 0.30. The *criticalThresh* parameter was then plotted against the *actChangeThresh* parameter and the performance results measured by both criteria are shown in figure 17. The best performance as measured by correct FOA—approximately 95%—is found at parameters values of 0.5 for *criticalThresh* and between 0.7 and 0.8 for *actChangeThresh*, as shown in figure 17a. The best task performance (100%) is found at parameter values of *criticalThresh* between 0.4 and 0.5 and *actChangeThresh* between 0 to 0.4 (figure 17b).

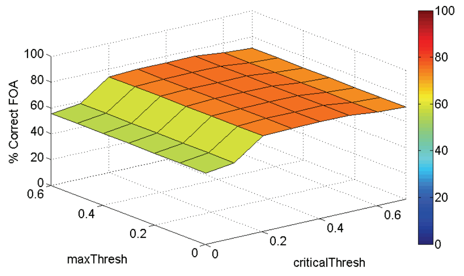
Although correct task switching is an important component of this system, selecting the correct FOA in the most frames is even more important, especially since the task performance is still excellent when the parameter values for optimal FOA selection are chosen. Also note that the *actChangeThresh* range was extended from



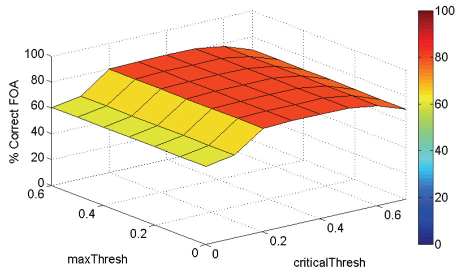
(a) actChangeThresh = 0.0



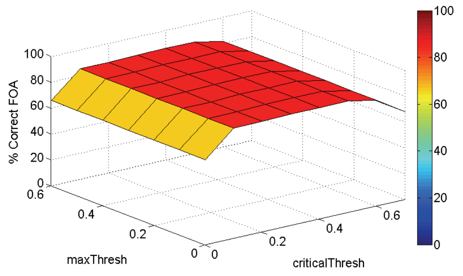
(b) actChangeThresh = 0.1



(c) actChangeThresh = 0.2

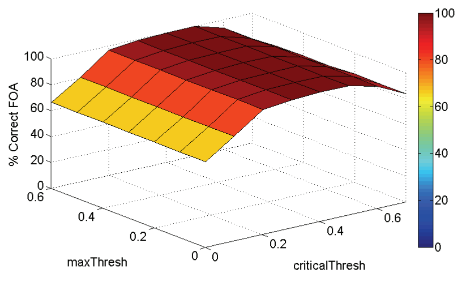


(d) actChangeThresh = 0.3

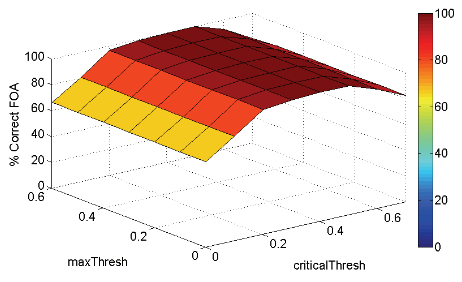


(e) actChangeThresh = 0.4

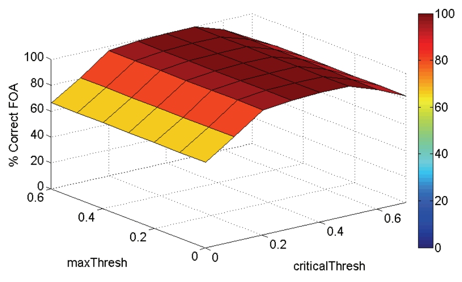
Figure 15: Performance measured by correct FOA criterion for varying parameter values



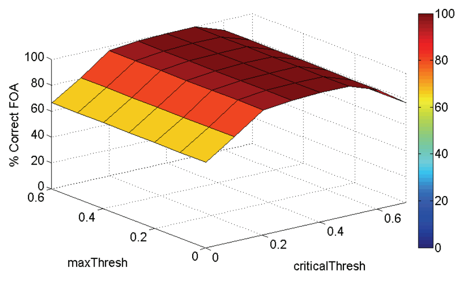
(a) actChangeThresh = 0.0



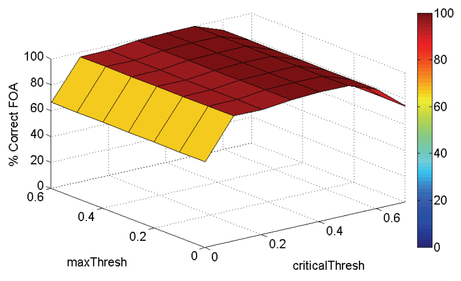
(b) actChangeThresh = 0.1



(c) actChangeThresh = 0.2

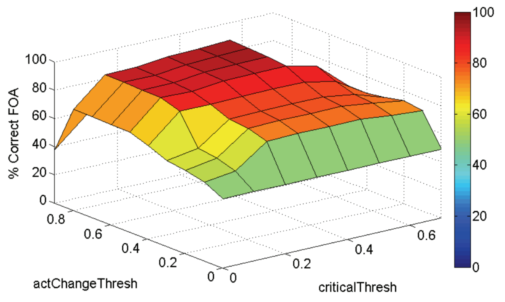


(d) actChangeThresh = 0.3

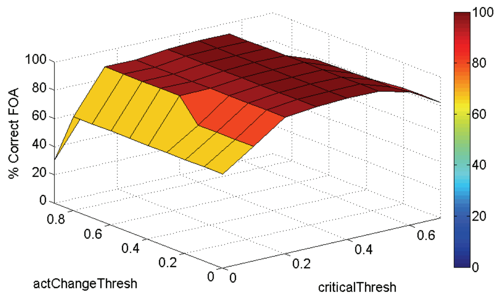


(e) actChangeThresh = 0.4

Figure 16: Performance measured by correct task criterion for varying parameter values



(a) % correct FOA



(b) % correct Task

Figure 17: Performance results measured by FOA and task for *maxThresh* = 0.3

0.4 to 0.9 when it was observed that best results were recorded on the edges of the tested range.

Since the result space is convex, a gradient ascent method [99] was then used to refine our estimate of the optimal parameter combination. The optimization algorithm was initialized with the best estimated parameter values, measured by the FOA criterion, from figure 17a: $maxThresh = 0.3$, $actChangeThresh = 0.70$, and $criticalThresh = 0.40$. After 10 iterations, the optimal parameters were found and are listed in table 3. It should also be noted that many parameter combinations in the neighborhood of the optimal parameter yielded high performance results. Therefore, system performance is not greatly affected by changes in parameter values on the order of ± 0.1.

Table 3: Optimal Parameter Values

Parameter Name	Value
maxThresh	0.423
actChangeThresh	0.855
criticalThresh	0.461

FOA Experiments

Real Data

Experiments were performed to determine the effects of increasing the number of distractors in the locale on system performance. The target was chosen to be a green, moving square and the distractors were identical but stationary green squares. (The distractors differed from the target in only one sensory modality: motion.) The distractors ranged in number from 0 to 8 and were arranged around the locale. The target was moved among these distractors without overlapping on top. Ten datasets for each number of distractors were recorded. Each trial had a length of 200 frames, which corresponds to 10 seconds of video at 20 frames-per-second (fps).

Each dataset was run through the system. The FOA nodes recorded for each frame were computed to calculate the performance of the system. The percentage of correct FOA was the criterion used to rate system performance. A “good” or “correct” FOA node is defined as any node overlapping over the target area. The average performance of the system for varying numbers of distractors is shown in table 4, where *FOA on distractor* represents the percentage of frames in which the FOA was on a distractor. The averages were also plotted with standard error bars in figure 18.

Table 4: Performance results for FOA experiments

Number of distractors	Correct FOA averages (%)
0	100
1	95.2
2	95.25
3	95.2
4	95.4
5	95.3
6	95.35
7	95.25
8	95.6

The performance of the system was high: the average percentage of correct FOAs never fell lower than 95% and the system attended to distractors a maximum of 14 frames out of 200.

Judging from this experiment, the performance of the system seems unaffected by an increase in distractors. Increasing the number of distractors, varying their placements, and allowing the target to sometimes overlap the distractors could yield stronger conclusions about the effects of distractor number on performance. We chose to perform additional trials using simulated data to explore larger distractor placement variations as well as a larger increase in the number of distractors. Before simulated data can be used, however, a benchmark simulation must be created to

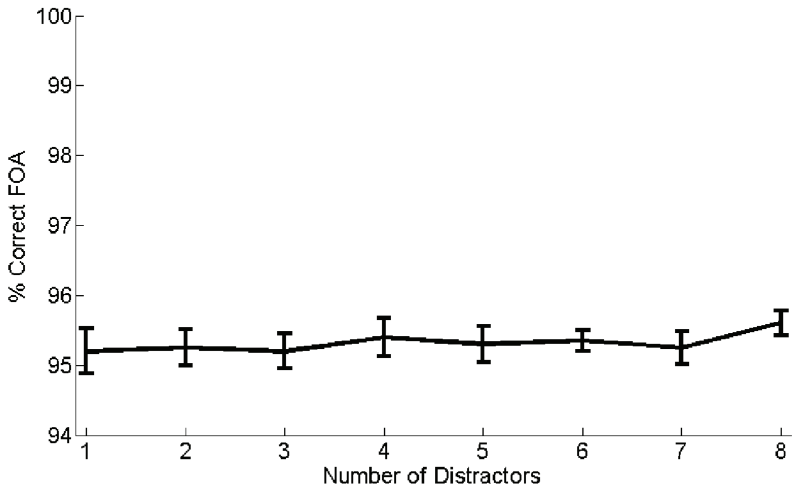


Figure 18: Percentage of correct FOA with respect to the number of distractors

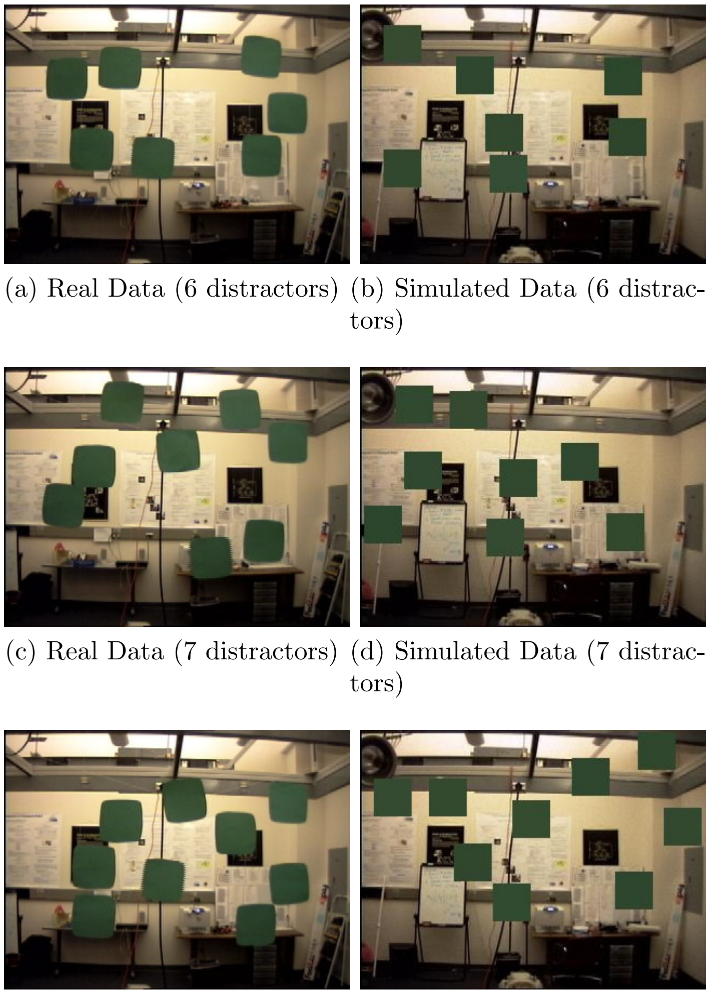
verify that the result trends of both the simulated data and the real world data are similar. Once this is shown, data can be simulated and multiple trials can be performed.

Benchmark Simulation

The real data used in the previous experiment was simulated so that the results of the real world dataset and simulated dataset could be compared. If the results have similar trends, it can be deduced that the system is behaving in the same way when operating on the simulated data and on the real data, and that simulated data can reliably be used to examine the effects of an increasing number of distractors on system performance.

To create the simulated data, 100 image frames of the locale without targets or distractors were recorded. The average background as well as variation from frame to frame were calculated. For each trial, the target and distractors' base color were sampled from a normal distribution centered around the mean target color and with standard deviation calculated from the real world data. The movement of the target was modeled to be similar to the real target's movement. Noise sampled from a normal distribution was added to the base motion so that target movement differed slightly from trial to trial, as was the case in the real datasets. Noise was also added to each target and distractors' pixels in each frame.

Figure 19 shows the real world data and the simulated data side by side.



(a) Real Data (6 distractors) (b) Simulated Data (6 distractors)

(c) Real Data (7 distractors) (d) Simulated Data (7 distractors)

(e) Real Data (8 distractors) (f) Simulated Data (8 distractors)

Figure 19: Examples of real data and corresponding simulated data

Ten trials for each number of distractors ranging from 0 to 8 were performed and the performance results were averaged at each number of distractors. The results of the experiment are plotted with the real data experiment results in figure 20.

Overall, the simulated data results are better than the real data results. The fact

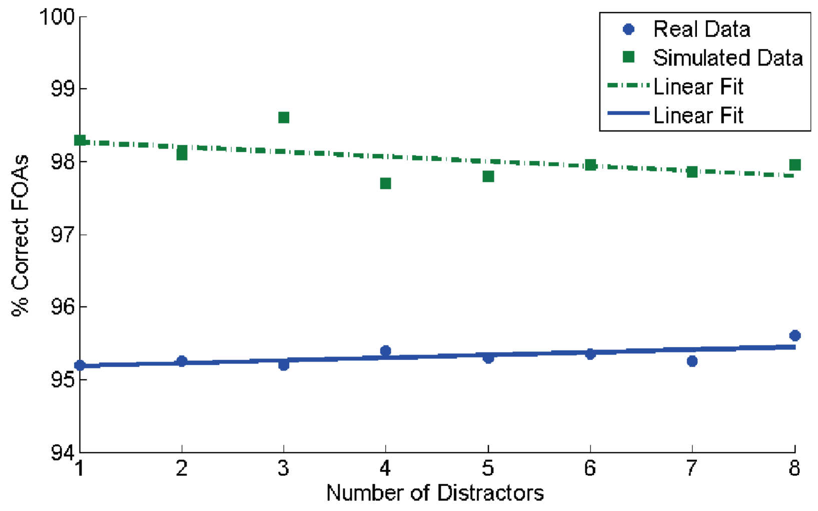


Figure 20: System performance comparison of real-world and simulated data

that target motion was better controlled in simulation accounts for this difference; the target was stationary more frequently in the real datasets and was, therefore, more frequently indistinguishable from the distractors. Visually, the trend lines are very similar, suggesting that the system behaves in the same way regardless of whether real or simulated data is used. The slopes were 0.038 and -0.067 for the real and simulated data respectively. An analysis of covariance was performed to determine whether the two data sets came from different distributions. The analysis results indicated that the data sets were not significantly different ($p > 0.13$). Therefore, we conclude that it is acceptable to use both real and simulated data to test the performance of the system.

FOA Experiment with Simulated Data

In this experiment, datasets were created with the number of distractors present in the locale varying between 0 and 10. Twenty trials for each situation were simulated and run; each trial having 211 image frames. The distractor locations were randomly generated from a uniform distribution spanning the image and were allowed to overlap.

For each trial, the target and distractors' base color were sampled from a normal distribution centered around the mean color derived from the real world data. Noise, sampled from a normal distribution centered around the frame-to-frame noise mean derived from the real data, was then added to these base colors in each frame and at each pixel independently.Target movement remained constant in each trial. Once the images and corresponding motion data were created, they were run through the system and the number of correct FOA was recorded.

Figure 21 shows the results averaged for each number of distractors. The standard error of the mean for each point is also shown. As can be seen, the performance of the system is at least 96% for all of distractor quantities.

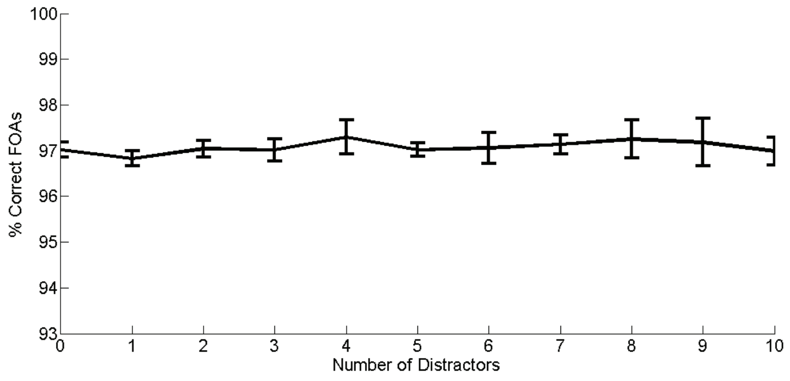


Figure 21: Average performance results for increasing number of distractors

A linear regression model was fitted to the data, as shown in figure 22, and the slope was calculated to be 0.018, which means that the trend line is nearly flat. This shows that the performance of the system is not affected by an increasing number of distractors that closely match the target. The system does not have much trouble identifying a target even though it may differ from distractors in only one sensory modality.

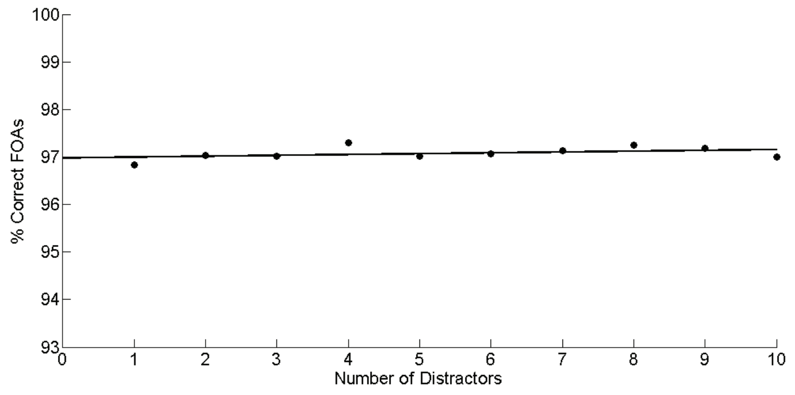


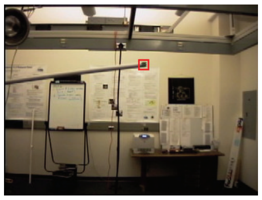
Figure 22: Average performance results for increasing number of distractors

Size Experiment

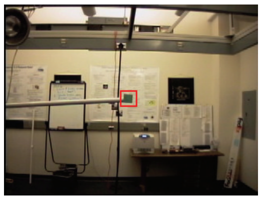
An experiment was performed to determine the effects of target size on system performance. A simulation was run where the target size decreased in each trial; the number of frames with a "correct" FOA was recorded in each trial. A correct FOA contains at least a portion of the target within the chosen node's area, as indicated on the image. Each dataset contained 185 frames. The target is a green moving square, and its size was originally 7 inches per side in the FOA experiments described in the previous section. The target size was decreased in 1-inch increments until it reached 4 inches per side, followed by 0.5-inch increments until the sides measured 1 inch. Table 5 lists the different target sizes used in each dataset and figure 23 shows an image of each target size. Obviously the target's apparent size fluctuates with its distance from the camera; therefore, the sizes are better represented by pixel area, as listed in the table. The percentage of the image as well as the percentage of the node occupied by the target are also listed. (Image sizes are 240 by 320 pixels which give an area of 76800 pixels. Average node sizes are 24 by 31 pixels; the area is 744 pixels.) The most important measurement for determining performance is not so much target size with respect to image size as it is the target size with respect to node size since this will determine the amount of activation present at the node.

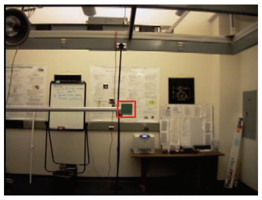
Table 5: Target Sizes

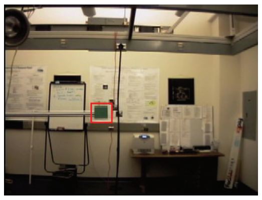
inches	pixels (approx.)	% of image	% of avg. node
1x1	5x5	0.03	3.36
1.5x1.5	8x8	0.08	8.6
2x2	11x11	0.16	16.26
2.5x2.5	15x15	0.29	30.24
3x3	17x17	0.38	38.84
3.5x3.5	19x19	0.47	48.52
4x4	21x21	0.57	59.27
5x5	28x28	1.02	105.3
6x6	33x33	1.42	146.4
7x7	38x38	1.88	194.1

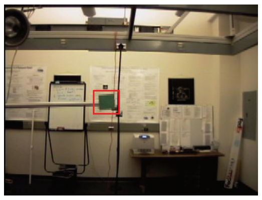

(a) 1in target

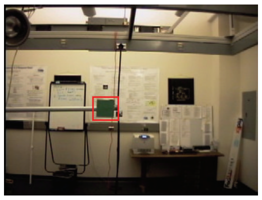
(b) 1.5in target

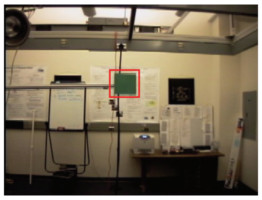

(c) 2in target

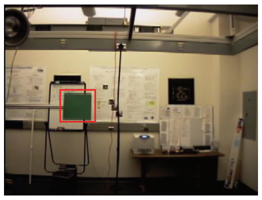

(d) 2.5in target


(e) 3in target


(f) 3.5in target


(g) 4in target


(h) 5in target


(i) 6in target

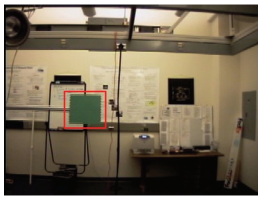

(j) 7in target

Figure 23: Variations in Target Size

The performance of the system was examined by plotting the number of correct FOA with respect to the target size in terms of the percentage of the node it covers, for a given SES tessellation, as shown in figure 24. The SES tessellation used in this experiment—and throughout this dissertation—was 14.

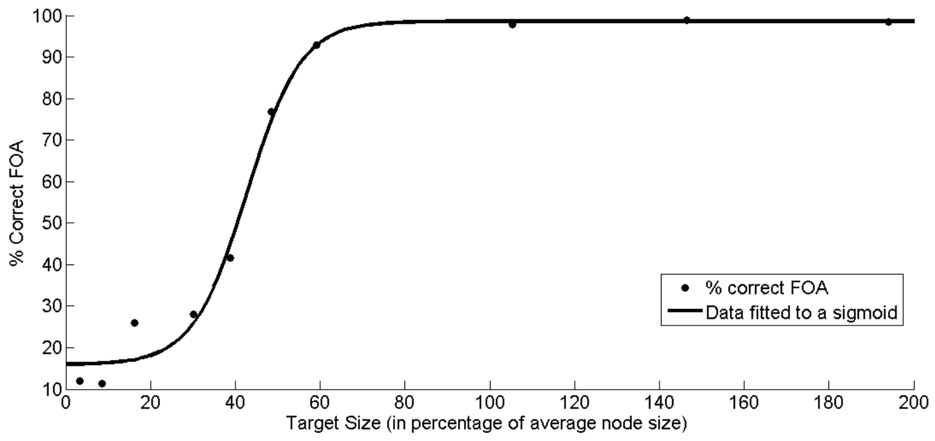


Figure 24: Percentage of correct FOA with respect to target size

As can be observed from figure 24, the data can be fitted nicely to a sigmoid function. In this case, a target size of 40% of the average node yields 50% performance results. When the target size is increased to approximately 50% of the node size, performance results become satisfactory (approximately 80% correct FOA). Therefore, we conclude that to have reliable system performance, the target size should be approximately at least 50% of the average SES node size for the given tessellation. Alternately, if the average target size is known in a locale, the SES tessellation should be chosen to yield nodes that are no larger than twice the average target size. To refine this size threshold, however, more trials would have to be performed with target sizes variations of a smaller step size. This could be done in simulation.

Task-Switching Experiments

An experiment was performed to observe the task-switching behavior of the system. A simple 2-task situation was created to first examine system performance under all possible task and priority combinations. A more-intensive situation involving multiple targets and tasks was then developed to push the boundaries of the system.

Simple Task Experiment

The system's task-switching behavior is dependent upon a few factors. It is affected by the real world data being fed into the system, the priority-level assigned to each defined task, and the initial task chosen before the start of the experiment and loaded onto the task stack. If multiple tasks are allowed to have the same priority, system performance could potentially also be affected by the order in which the tasks are listed.

This experiment was designed to investigate all of these conditions. Four real-world situations were created involving a red square target and a green square target; sensor data was collected from each situation. The different factor in each situation was the motion of the target(s). In the first data set, both the red and green targets are stationary; similarly, both targets are moving in the fourth data set. The second and third sets alternate with one moving target and the other stationary.

For each of the data sets above, we have four possible task priority combinations that could be active in the system: the first two combinations are the red task having higher priority than the green task and the green task having higher priority than the red task. Both tasks could have the same priority; if so, either the red one or the green one will be listed first. This constitutes the last two combinations. Furthermore, there is always an initial task chosen prior to the start of the simulation. For each of the four task priority combinations, the initial task could either be the red or green task, yielding a total of 8 different test conditions for each of the four original data set. Table 6 summarizes these possible testing conditions. As can be seen, there are a total of 32 different test conditions.

Table 6: Test conditions for task experiment

Data Set	Priority							
	Equal Priority (Red Listed First)		Equal Priority (Green Listed First)		Green Higher Priority		Red Higher Priority	
Red and Green Targets Stationary	Initial Task RED	Initial Task GREEN	Initial Task RED	Initial Task GREEN	Initial Task RED	Initial Task GREEN	Initial Task RED	Initial Task GREEN
Red Target Stationary, Green Target Moving	Initial Task RED	Initial Task GREEN	Initial Task RED	Initial Task GREEN	Initial Task RED	Initial Task GREEN	Initial Task RED	Initial Task GREEN
Red Target Moving, Green Target Stationary	Initial Task RED	Initial Task GREEN	Initial Task RED	Initial Task GREEN	Initial Task RED	Initial Task GREEN	Initial Task RED	Initial Task GREEN
Red and Green Targets Moving	Initial Task RED	Initial Task GREEN	Initial Task RED	Initial Task GREEN	Initial Task RED	Initial Task GREEN	Initial Task RED	Initial Task GREEN

The results of this experiment are shown in table 7. Each data set contained 191 frames. For each set of test conditions, the system performance is measured by both the FOA criteria in the first result row of each dataset and the task criteria in the second result row of each dataset. Results are given in percentage of frames spent of either the green or the red target or task (abbreviated R for red and G for green), depending on which task had higher priority, for ease of comparison.

Table 7: Task Experiment Results

		Priority							
		Equal Priority (Red Listed First)		Equal Priority (Green Listed First)		Green Higher Priority		Red Higher Priority	
Data Set		Initial Task RED	Initial Task GREEN	Initial Task RED	Initial Task GREEN	Initial Task RED	Initial Task GREEN	Initial Task RED	Initial Task GREEN
Red and Green Targets Stationary	FOA	100% R	100% G	100% R	100% G	100% G	100% G	100% R	100% R
	Task	100% R	100% G	100% R	100% G	100% G	100% G	100% R	100% R
Red Target Stationary, Green Target Moving	FOA	80.6% R	100% G	80.6% R	100% G	100% G	100% G	80.6% R	80.6% R
	Task	100% R	100% G	100% R	100% G	100% G	100% G	100% R	100% R
Red Target Moving, Green Target Stationary	FOA	100% R	79.6% G	100% R	79.6% G	79.6% G	79.6% G	100% R	100% R
	Task	100% R	100% G	100% R	100% G	100% G	100% G	100% R	100% R
Red and Green Targets Moving	FOA	81.2% R	81.2% G	81.2% R	81.2% G	81.2% G	81.2% G	81.2% R	81.2% R
	Task	100% R	100% G	100% R	100% G	100% G	100% G	100% R	100% R

This data allows us to examine the system's behavior under a variety of states. The first observation made from table 7 is that the order in which equal priority tasks are listed in the priority-ordered task list does not affect performance. This is shown by noting that the two sub columns (*initial task Red* and *initial task Green*) of the *Equal Priority (Red listed first)* column are identical to the two sub columns of the *Equal Priority (Green Listed First)* column.

What does play a role, however, is the initial task selection. It was observed that when both tasks have equal priority, the initial task dictates which of the two tasks will be active for the entirety of the experiment. If both targets are stationary, it also dictates which of the targets the FOA will be on. If one or both targets is moving, however, and the moving target is not selected as the initial task, the FOA will be distracted away from the target indicated by the initial task approximately 20% of the time.

When one task is given higher priority than the other, the initial task selection does not affect system performance at all. This is because we always push all tasks having a higher priority than the initially selected task onto the task stack at the beginning of an experiment to detect situations where information matching a higher-priority task could be active (and static) in the locale. It was also observed that the active task in the system is the higher-priority in 100% of the frames, regardless of any motion occurring in the scene. Although the task remains fixed, the FOA does shift to the lower-priority target when that target is moving. Distracting FOA shifts occur approximately 20% of the time when either task is given highest priority.

Benchmark Experiment with Simulated Dataset

The above experiment was designed to examine the system's general task-switching behavior. We would also like to determine whether increasing the number of tasks (and, therefore, targets) has any impact on system performance.

We chose to use simulated data for this particular experiment to allow us to create datasets with more tasks entering and leaving the locale as well as greater control over target motion than would have been possible with real-world data. A benchmark simulation was performed for the number of distractors experiment, and it was found that the FOA selection behaved similarly with either real world or simulated data. However, the conditions under which the system was tested did not include multiple tasks or any task-switching behaviors. Therefore, we must first determine whether the task-switching behavior is consistent under both real and simulated data. Once this is shown, experiments involving task-switching can then be performed with simulated datasets.

To compare system behavior for real and simulated data, a dataset was created to closely correspond to the second real world dataset used in the above simple task experiment. This dataset involves a red stationary target and a green moving target. The average background as well as variation from frame to frame were calculated to keep the same amount of noise in the simulated data than was present in the real world data. These statistics were used to generate the simulated background. Each target's mean color, standard deviation, location, and size were also recorded and used to create the simulated dataset. For each frame, the color value of each target pixel was sampled from a Gaussian distribution centered at the target's mean color and with standard deviation calculated from the real world data. The movement of the target was derived from the real dataset by finding the location of the target's centroid in each frame. Figure 25 displays corresponding frames from the real and simulated datasets for illustrative purposes.

To compare system behavior under real and simulated data, the simulated data was run through the 8 same test conditions detailed in table 7 for that dataset. These tests exhaustively represent all possible combinations of starting conditions for two

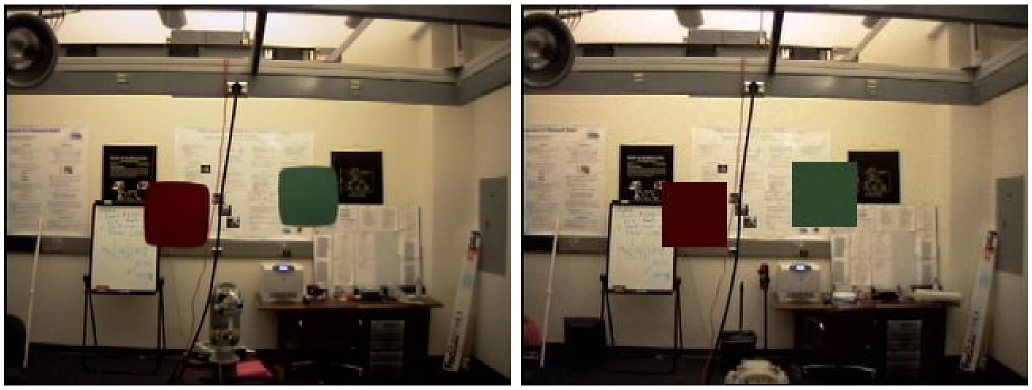

(a) Frame 104: Real Data (b) Frame 104: Simulated Data

Figure 25: Corresponding frames of the real (left) and simulated (right) dataset

targets. The results of this experiment are shown in table 8. For ease of comparison, the results obtained with real data are reproduced in this table as well.

Table 8: System performance results of real-world and simulated datasets.

		Priority							
		Equal Priority (Red Listed First)		Equal Priority (Green Listed First)		Green Higher Priority		Red Higher Priority	
Data Set		Initial Task RED	Initial Task GREEN	Initial Task RED	Initial Task GREEN	Initial Task RED	Initial Task GREEN	Initial Task RED	Initial Task GREEN
Real World Dataset	FOA	80.6% R	100% G	80.6% R	100% G	100% G	100% G	80.6% R	80.6% R
	Task	100% R	100% G	100% R	100% G	100% G	100% G	100% R	100% R
Simulated Dataset	FOA	85.3% R	100% G	85.3% R	100% G	100% G	100% G	85.3% R	85.3% R
	Task	100% R	100% G	100% R	100% G	100% G	100% G	100% R	100% R

As can be observed from table 8, the system performance is very similar for the real world and simulated datasets. The only difference is a 5% increase in performance for the simulated set when the system should be focused on the red target. This can be attributed to the slight differences between the real and simulated datasets, and is not enough evidence to discredit the use of simulated data to observe system behavior. Therefore, the following task experiments will be performed using simulated data.

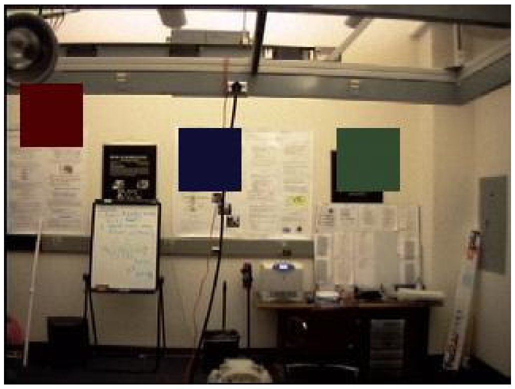
Task Experiment with Simulated Dataset

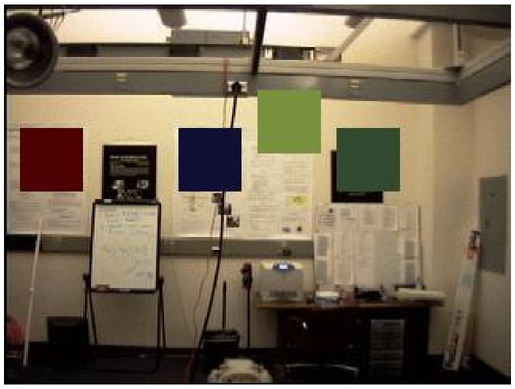
This experiment was performed to discover the effects of system performance when there are an increasing number of possible tasks. A simulation was created in which targets representing different tasks enter and leave the locale. The targets were chosen to be squares of different colors and separate datasets were created for target totals ranging between 2 and 6. The mean color for each target was calculated from sequences of real images as described in the previous section. In each simulated image, the color of each target pixel was sampled from a Gaussian distribution centered at the mean target color and with a standard deviation derived from the real world images.

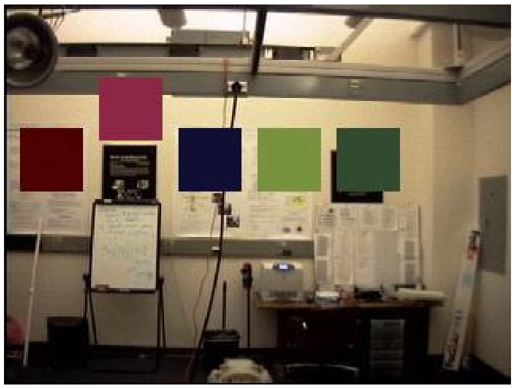
Datasets were created where targets would enter the locale from above one at a time; the targets would then exit in the order that they came in. The number of frames that a target should be selected as FOA in each simulation was kept constant across all targets for ease of comparison. To determine whether the order in which targets enter the locale affects task and FOA selection, the priorities of each target present in the simulation was alternated. That is to say that each target was chosen as having the highest priority in its own trial. The order of entry of the targets into the locale was kept constant for each simulation. What was varied was the priority of each target. Table 9 summarizes the datasets used in this experiment and figure 26 contains frames from each dataset showing the targets used. Note that target order does not refer to the priority of each task matching the targets but instead it refers to the order in which the targets enter (and exit) the locale. Also, note that the number of targets and the number of tasks are the same in this experiment, and are used interchangeably since each target has a matching task.

The percentage of frames that a task should be active—and also that a target should be selected as FOA—were recorded. These statistics were recorded in table

(a) Dataset 1 (2 targets)


(b) Dataset 2 (3 targets)


(c) Dataset 3 (4 targets)


(d) Dataset 4 (5 targets)

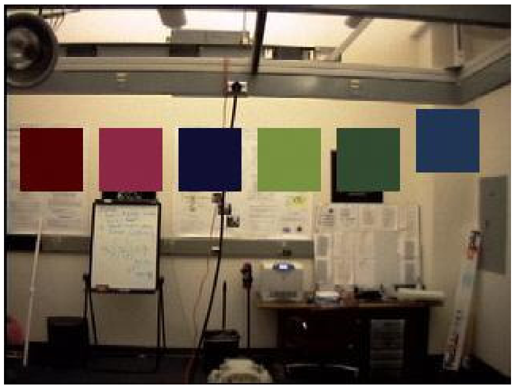

(e) Dataset 5 (6 targets)

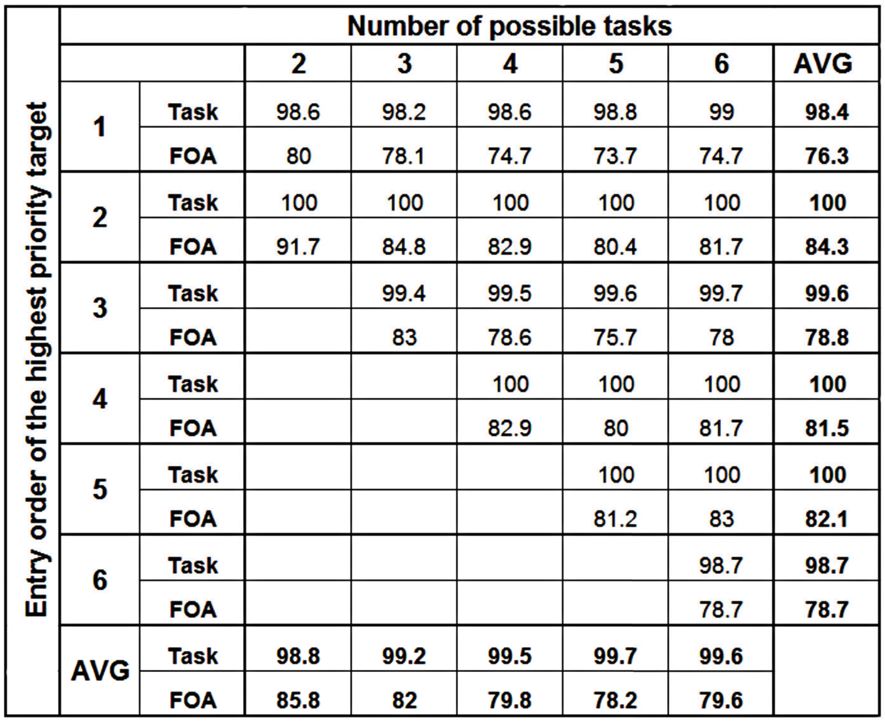
Figure 26: Frames from each simulated dataset.

Table 9: Simulated datasets

Dataset	Targets	Target Order
1	2	Red, Green
2	3	Red, Green, Blue
3	4	Red, Green, Blue, Lime
4	5	Red, Green, Blue, Lime, Pink
5	6	Red, Green, Blue, Lime, Pink, Sky Blue

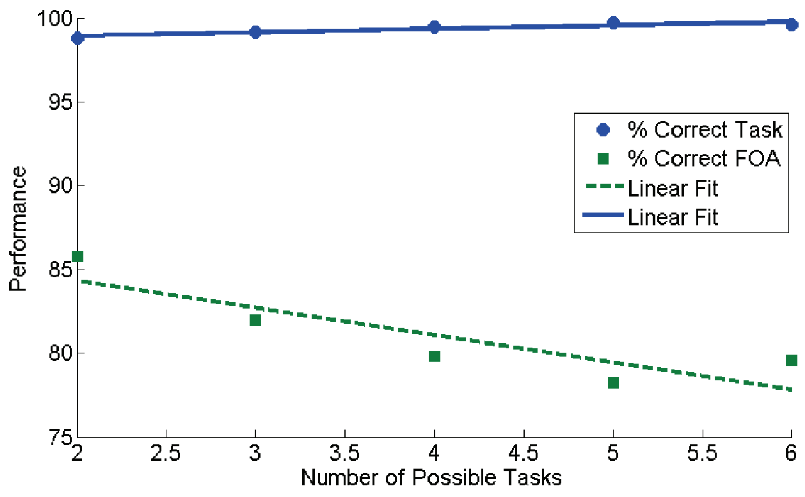
10. In this table, the columns represent the total number of tasks (and targets) in each dataset. The rows represent which of the tasks was given the highest priority. Some of the table's cells are left blank since we only have as many trials as there were targets in each dataset (there is no data for task 3 having the highest priority for the dataset with only 2 tasks). Each row and each column was then averaged. The column averages give information on the effect of increasing the number of total possible tasks while the row averages give information on the effect of the order in which the highest priority task enters the locale.

Table 10: System performance with an increasing number of tasks.

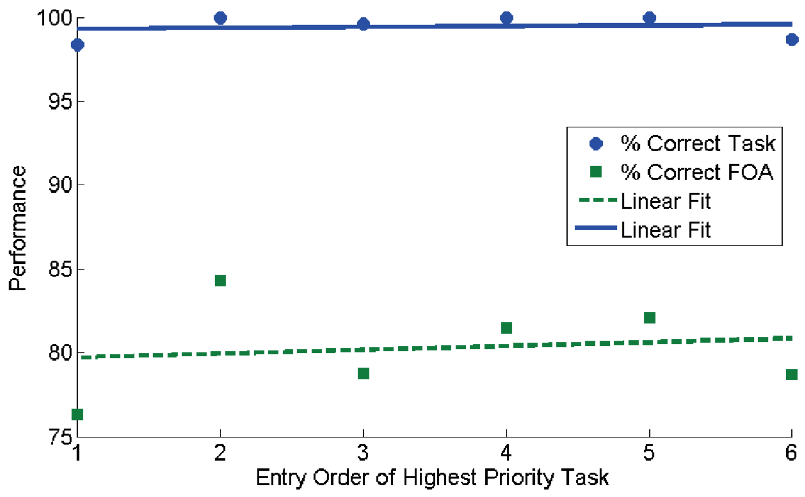


Entry order of the highest priority target		Number of possible tasks					
		2	3	4	5	6	AVG
1	Task	98.6	98.2	98.6	98.8	99	98.4
	FOA	80	78.1	74.7	73.7	74.7	76.3
2	Task	100	100	100	100	100	100
	FOA	91.7	84.8	82.9	80.4	81.7	84.3
3	Task		99.4	99.5	99.6	99.7	99.6
	FOA		83	78.6	75.7	78	78.8
4	Task			100	100	100	100
	FOA			82.9	80	81.7	81.5
5	Task				100	100	100
	FOA				81.2	83	82.1
6	Task					98.7	98.7
	FOA					78.7	78.7
AVG	Task	98.8	99.2	99.5	99.7	99.6	
	FOA	85.8	82	79.8	78.2	79.6	

The row averages and column averages were also plotted for both the correct task criterion and the correct FOA criterion. A linear regression was applied to the data, as can be seen in figure 27.



(a) Increasing number of tasks



(b) Varying entry order of the highest-priority task

Figure 27: System performance result trends for multiple tasks.

As can be observed from the graphs, there is no noticeable trend measured by either method indicating that the order in which a higher priority task enters the locale would negatively affect system performance (slopes were 0.05 and 0.23, respectively). Moreover, the number of tasks active in a locale does not affect system performance when measured by the correct task criterion (slope was 0.21). There is, however, a downward trend when performance is measured by the correct FOA criterion: the slope of this regression was calculated to be -1.62. This leads us to believe that the number of tasks active in a locale does negatively affect system performance. This is an intuitive results since the more targets matching different tasks are present in the locale, the more likely the FOA is to focus on one of them. This is especially true if the targets matching lower priority tasks are all moving: this motion will attract the FOA and temporarily shift it away from the highest priority task.

Habituation Experiments

To examine the effects of habituation, a simple simulated dataset was created. This dataset is very similar to the second real world dataset used in the simple task experiment described previously, and involves a red stationary square and a green moving square. The green target moves to the left or right for one frame every 5 frames (on frame numbers ending in 1 and in 6). The task was to attend to the red target, but the movement of the green distractor attracts the FOA onto itself. A distractor node was identified and monitored in each frame so that the habituation variable values can be observed. Figure 28 shows a frame from the dataset with the distractor node selected as FOA. Note that the distractor node is not centered on the green square because the motion only appears on the edges.

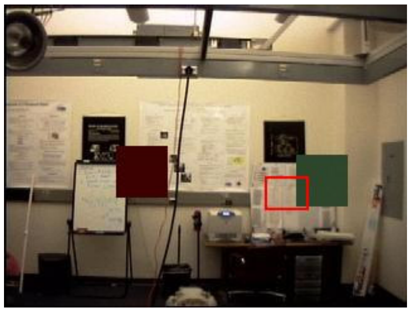
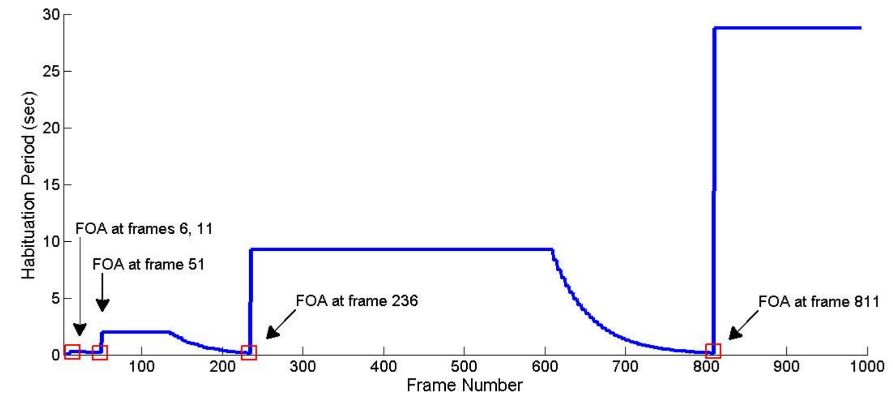


Figure 28: Dataset frame with distractor node selected as FOA.

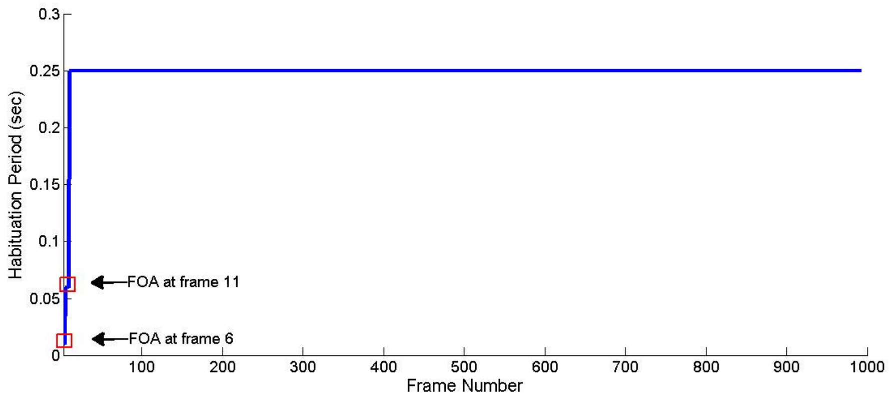
There were a total of 990 frames in this dataset, and the distractor node was selected as FOA 5 times. (Another distractor node was selected in 5 frames as well, for a total of 10 frames (5.1% of frames with distractor motion) in which the FOA was on the distractor target.) This is a tremendous improvement in comparison to 198 frames (100% of frames with distractor motion) on the distractor when habituation is not used in the system.

The habituation values for the distractor node are plotted in figure 29. The node's *HabituationPeriod* variable (abbreviated t_{hab}) is first initialized to 0.01 seconds, which is a value less than the posting rate of the fastest SPM. The distractor node then becomes the FOA at frame 6, as indicated on the graph. Once this occurs, t_{hab} is modified: since the node has not been FOA in the past, a predefined time amount corresponding to the time between two consecutive postings (0.05 seconds) is added, for a total of 0.06 seconds. The distractor node is then selected as FOA again at frame 11. The time difference between frame 11 and frame 6 is then used to modify t_{hab}; this difference is 0.25 seconds. This value prevents the FOA from shifting to the distractor node in frames 21, 26, etc., when the green distractor moves. As described in chapter III, t_{hab} is decayed back to its initial value over time so that the system can detect changes in a distractor's periodicity. The decay begins after a period of time equal to one habituation period plus a recovery period calculated with the current t_{hab} and t_{rec} values. Because of this decay, the distractor node becomes FOA again at frame 21. The difference between frame 51 and frame 11 (the previous FOA node) is then used to calculate t_{hab}, yielding a value of 1.77 seconds. The distractor node is selected as FOA again at frames 236 and 811 once t_{hab} undergoes other decays. Note that the time between FOA selection is increasing. The distractor node is not selected again as FOA.

Habituation causes the distractor to shift the FOA less. Additionally, the time period between the node being chosen as FOA is increasing. This is because of the use of the last time the node was FOA in the habituation variable calculations as



(a) With decay



(b) Without decay

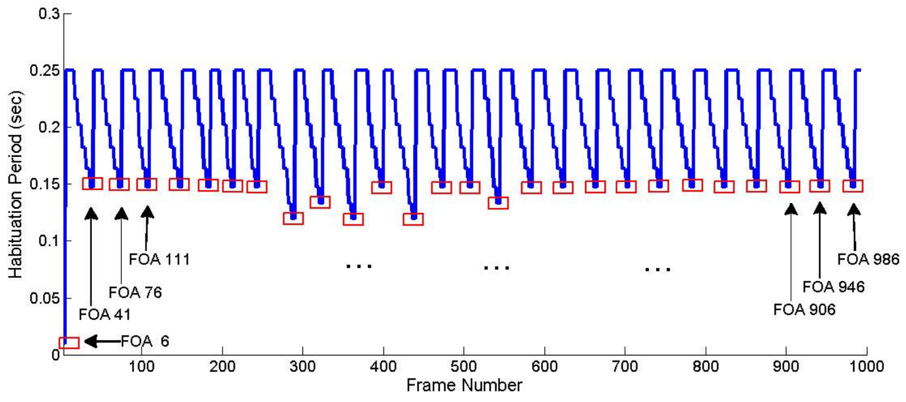
Figure 29: *HabituationPeriod* values for distractor node using the lastFOA modification method.

well as the decay of the variable. This is fine for applications where changes in the periodicity of the distractors is not important; in fact, if a distractor never needs to be attended again, the habituation variable decay can be omitted. The performance results would then look like those of figure 29b.

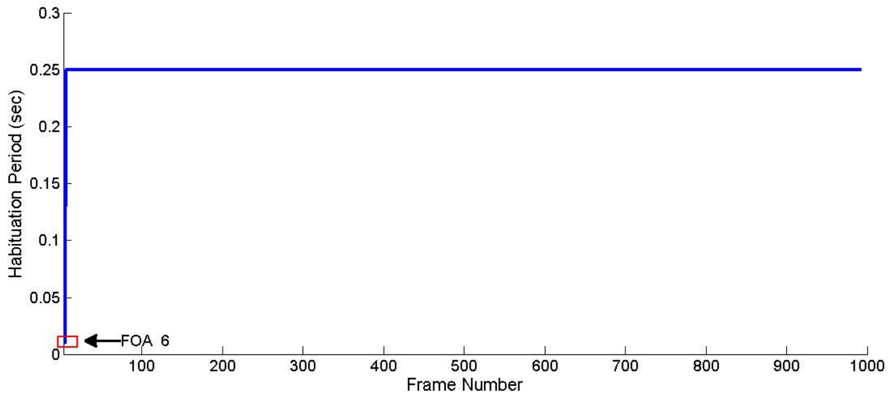
Some applications may necessitate the ability to attend to the changing periodicity of a distractor. Although the system as described above is capable of doing so, it will take a progressively longer time period to come back and attend to the distractor as the time between FOA increases, thereby increasing the time it takes for the *habituationPeriod* to decay back to its initial value. In this case, it may be necessary to modify the habituation algorithm so that the time difference between consecutive identical postings is used to set the value of t_{hab} and t_{rec} instead of the time since the node was last the FOA.

The simulated dataset was run through the system modified as described above and the results are plotted in figure 30a. Distractor nodes caused a total of 57 FOA shifts from the red target (28.8% of frames with distractor motion). Although the time between postings method yields more frames with a distractor FOA than the last FOA method, the performance is still much better than without any habituation yet it allows some monitoring of the locale on a regular interval. It can be seen as a middle ground between complete habituation of a distractor and no habituation at all. If, however, the habituation variables are not allowed to decay back to their original value, as was shown for the last FOA method in figure 29b, the distractor is completely habituated and the FOA selector does not select it again after the initial selection at frame 11 as shown in figure 30b, and the performance is equivalent to the last FOA method without decay.

Figure 31 shows the number of FOAs when the last FOA method and the time difference method of applying habituation are used as well as when no habituation is used.

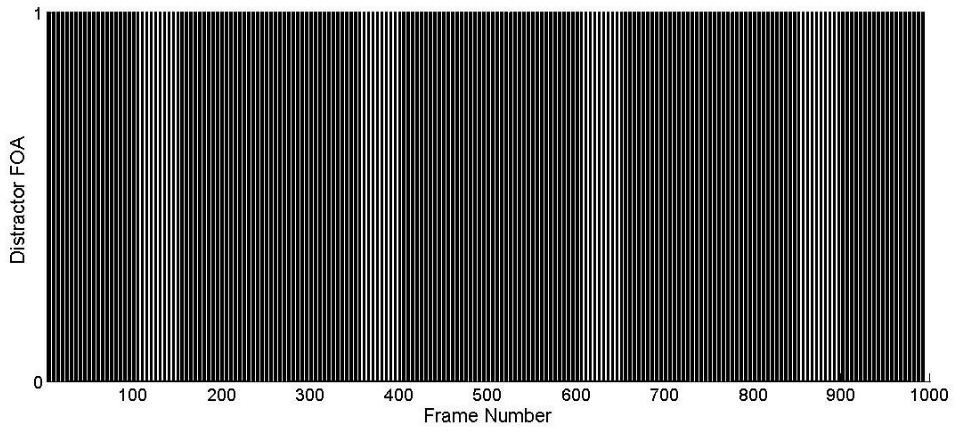


(a) With decay

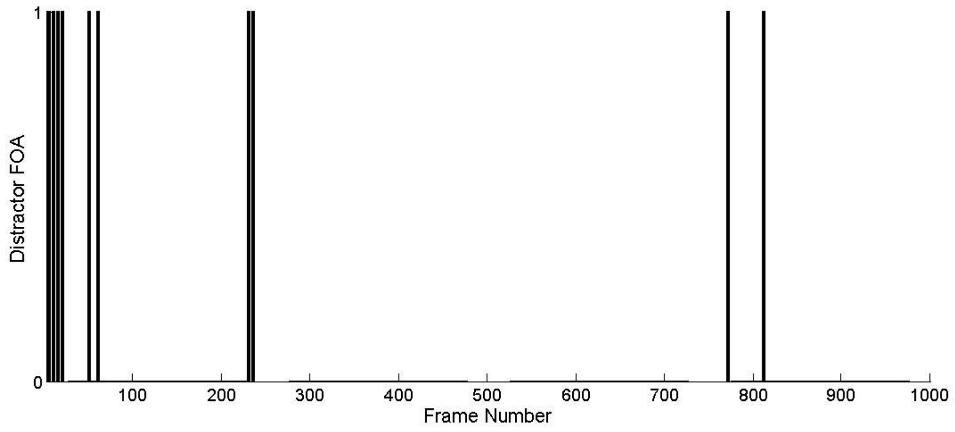


(b) Without decay

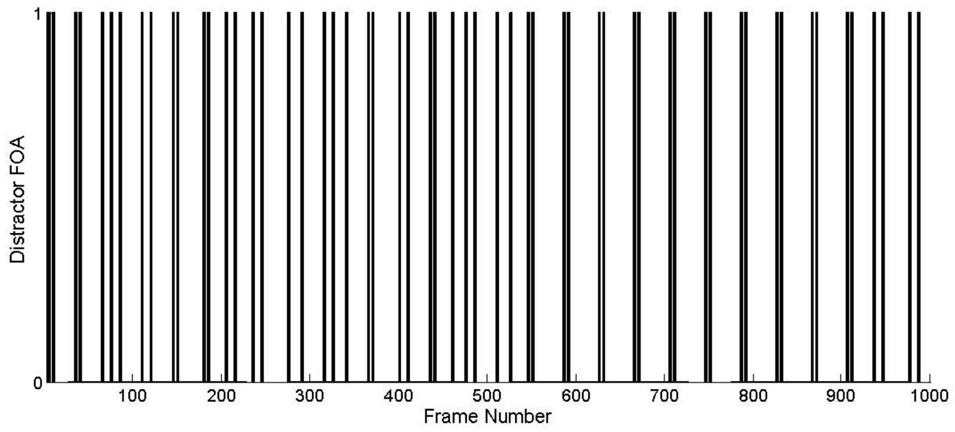
Figure 30: *HabituationPeriod* values for distractor node using the time between postings modification method.



(a) Without habituation



(b) Last FOA method



(c) Time difference method

Figure 31: Frames where FOA was on a distractor node under no habituation and habituation using last FOA and time difference methods.

The last FOA method and the time between posting method will be used in the experiment to detect a periodic distractor with both constant and changing periodicity described in a later section of this chapter.

Periodic Distractor Experiments

It has been shown in previous experiments that system performance is not significantly changed when simulated data is used instead of real data. In this set of experiments, we wish to observe the effects that a periodic distractor has on system performance. We chose to simulate the data for these experiments to have greater control over the distractor presentation intervals, either by keeping it fixed or by randomly varying it. The same method of simulated data generation as was used in the previous experiments was used here.

Each dataset consists of 990 image frames (about 50 seconds) containing a red target and a green (moving) distractor. Several methods of distractor presentations are used here. In the first dataset, the green distractor is moved in place constantly while in the second dataset, the distractor is moved on a regular interval. This interval is set at 10 frames, which corresponds to 0.5 seconds. To determine whether a regular interval is necessary for good system performance, another dataset is created in which the distractor is moved on a randomly generated interval sampled from a uniform distribution. The time period between motion ranges between 0.5 and 1 second (10 to 20 frames). Such a non-fixed interval is more like the "fixed" interval that could be achieved with real world data.

The last two datasets were created to determine whether the system can detect changes in the periodicity of the distractor. Dataset 4 begins with a distractor moving on a 0.5 second interval. After 6 sightings, the interval increases to 1 second. Dataset 5 is similar to dataset 4, except that the interval begins at 1 second and decreases to

0.5 seconds. In both cases, the time it takes for the system to notice the change will be calculated. Table 11 summarizes each dataset.

Table 11: Datasets for periodic distractor experiments

Dataset	Name	Description
1	Constant motion	Movement in place in every frame
2	Fixed interval	Periodic motion on regular interval every 0.5 seconds (10 frames)
3	Random interval	Periodic motion on randomly changing interval between 0.5 and 1 second (10 to 20 frames)
4	Increasing interval	Periodic motion on regular interval starting at every 0.5 sec (10 frames) for 10 presentations, then interval is repeatedly doubled and held for 10 presentations.
5	Decreasing interval	Periodic motion on regular interval starting at every 2 seconds (40 frames) for 10 presentations, then interval is repeatedly halved and maintained for 10 presentations.

Results from these experiments are shown and discussed below. Note that the percentages reflect the number of frames in which the distractor was selected as FOA with respect to the number of frames where the distractor was moving. In the constant motion experiment, the distractor was moving in all 990 frames, but in the case of periodic distractor experiments, this number is significantly less.

Constant Motion Experiment

In this experiment, the distractor was constantly moving in place, triggering FOA shifts to distractor nodes. Table 12 shows results with no habituation and habituation applied with both the time difference and last FOA methods.

The FOA shifted to the distractor in 37.6% of the frames when no habituation was used in the system. Both habituation methods do improve system performance compared to having no habituation at all in the system. In this case however, the time difference method—shown in figure 32b for both distractors nodes—performs

Table 12: Constant motion experiment results

Method	Number (percentages) of frames on distractor
No Habituation	372 (37.6%)
Habituation (last FOA method)	12 (1.2%)
Habituation (time difference method)	81 (8.2%)

worse than the last FOA method. This is because motion occurs in every frame and the time between postings is so small that it decays back to its original value quickly, triggering another FOA shift onto the distractor. The FOA is on the distractor 8.2% of the time, which is still much better than the system operating without habituation. The last FOA method—shown in figure 32a—does a great job of habituating to the constantly moving distractor: the FOA shifted to the distractor in only 1.2% of the frames.

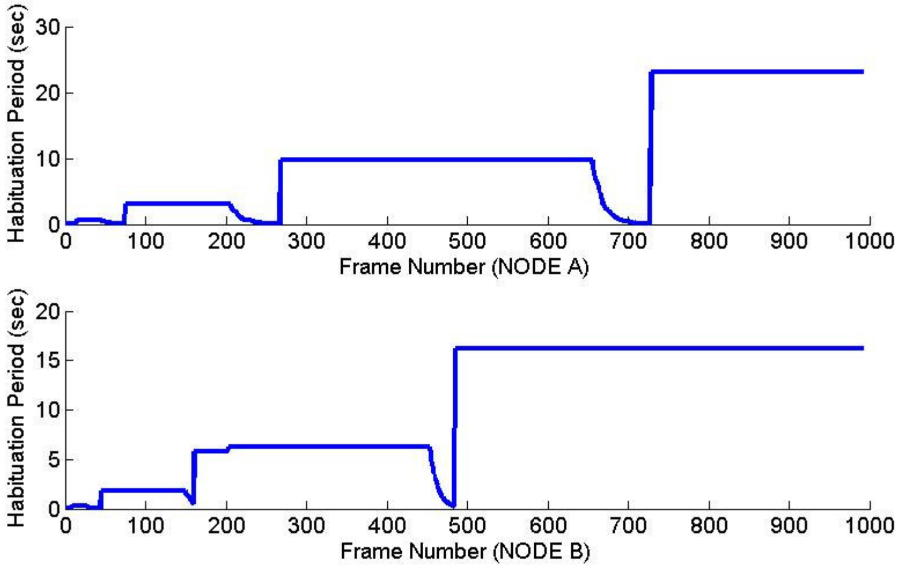
Fixed Interval Experiment

This experiment was performed to observe the habituation effects on a distractor occurring on a fixed interval. Since motion occurred regularly every 10 frames, the FOA could shift to the distractor node in 99 frames. When there was no habituation present in the system, the FOA shifted to the distractor each time it moved (100% of the frames with distractor motion); this was greatly improved with the time difference and last FOA methods of applying habituation, as shown in table 13.

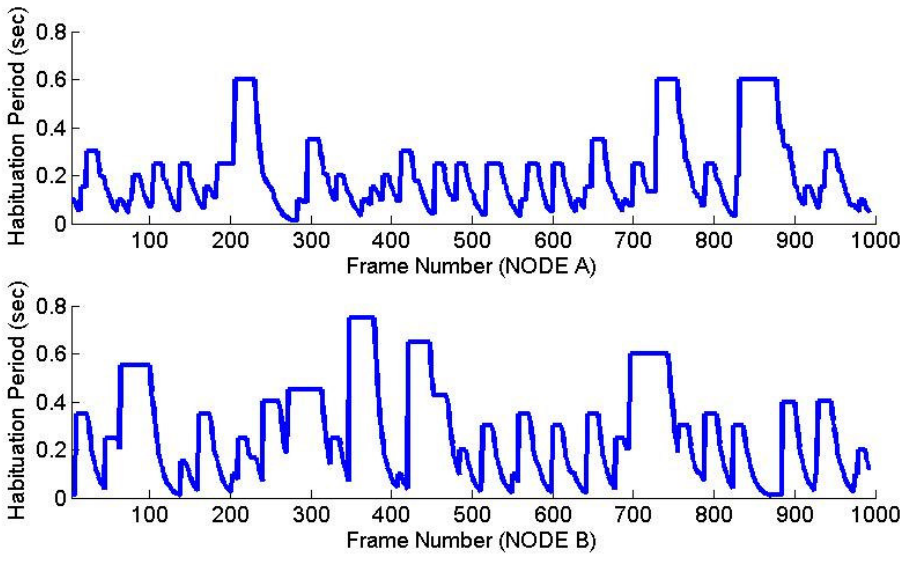
Table 13: Fixed interval motion experiment results

Method	Number (percentages) of frames on distractor
No Habituation	99 (100%)
Habituation (last FOA method)	8 (8%)
Habituation (time difference method)	30 (30%)

Performance improvements are greater over time since the habituation period is modified mostly in the beginning of the set: for the last FOA method, the first two



(a) Last FOA method



(b) Time difference method

Figure 32: *HabituationPeriod* values for distractor nodes for the constant motion experiment.

occurrences serve to habituate node B and the second occurrences habituate node A, as shown in figure 33a.[11] Subsequently, FOA shifts only occur at frames 100 and 470 and frames 120 and 440 for node B and node A respectively.

When the time difference method of applying habituation is used, the first two occurrences serve to habituate nodes B and A using the time between postings. Subsequent FOA shifts occur regularly at each node, as shown in figure 33b. As mentioned previously, if the time difference method is used, FOA shifts will occur on a shorter interval as the habituation variables decay to their initial values. For the last FOA method however, FOA shifts will occur farther and farther apart as the time between the node is chosen as FOA increases. Therefore, the last FOA method may be a better fit when distractors occur repeatedly on a fixed interval unless the application necessitates that distractor locations be attended regularly, albeit not as often as they would be attended under no habituation.

Random Interval Experiment

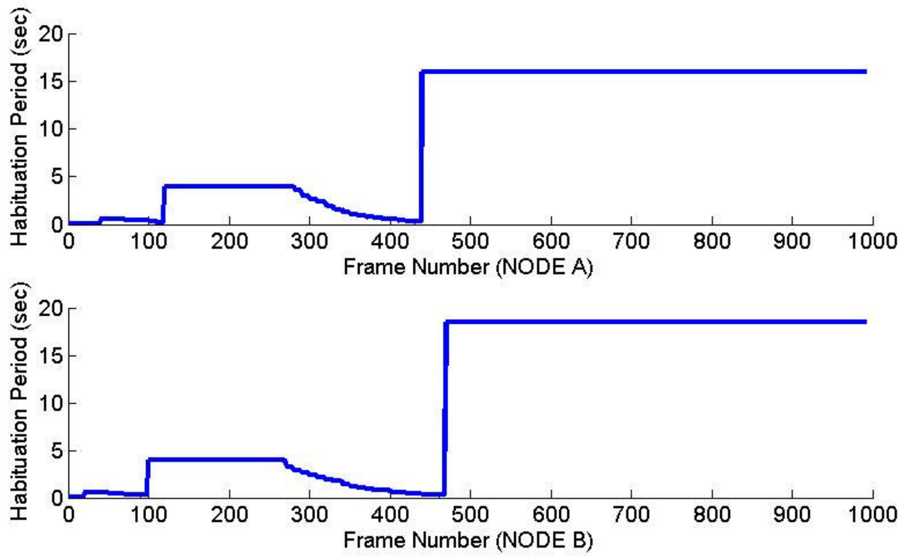
In this experiment, the interval between distractor motion randomly varied between 10 and 20 frames. Motion occurred in 64 frames, and each frame was selected as FOA when the system operated without habituation, as shown in table 14.

Table 14: Random interval motion experiment results

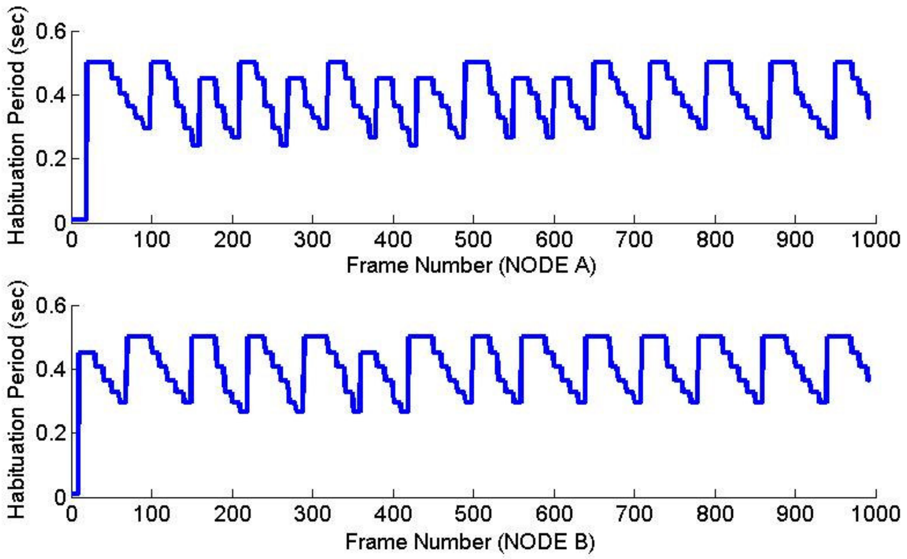
Method	Number (percentages) of frames on distractor
No Habituation	64 (100%)
Habituation (last FOA method)	8 (12.5%)
Habituation (time difference method)	16 (25%)

There were no decrease in system performance observed between this experiment

[11] Since nodes A and B are not adjacent (there is one two between them), their habituation variables are not spread to each other and the locations are habituated separated. If they were adjacent, however, the first two occurrences would habituation both locations and there would not be a FOA shift in frames 30 and 40.



(a) Last FOA method



(b) Time difference method

Figure 33: *HabituationPeriod* values for distractor nodes for the fixed interval experiment.

and the fixed interval experiment under the last FOA habituation method (figure 34a). The last FOA method performed exactly the same and habituation applied using the time difference method (figure 34b) performed even better because the longer period between distractor sightings caused longer habituation periods and decays. This experiment shows that the interval of distractor presentation does not have to be exactly the same to enjoy the performance benefits of the habituation subsystem.

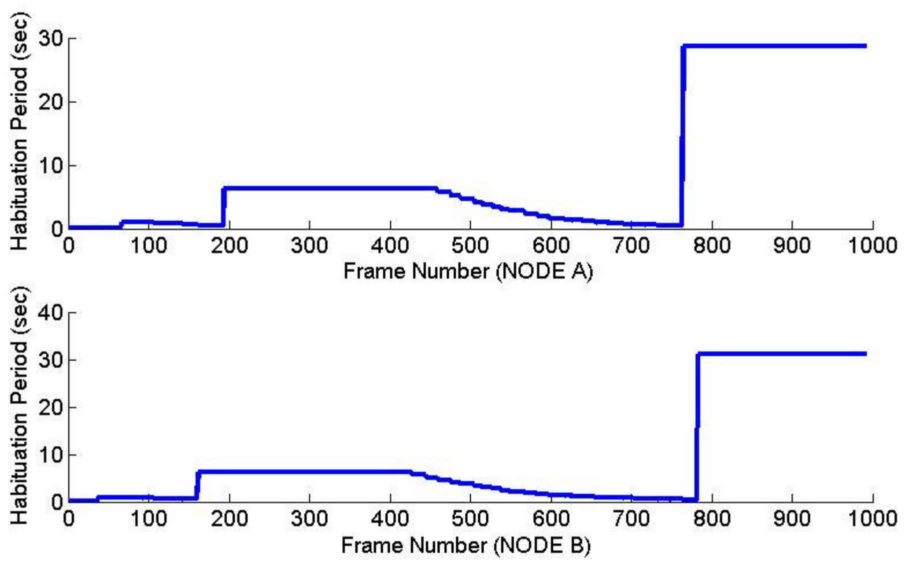
Increasing Interval Experiment

In this experiment, we want to observe the amount of time it takes for the system to notice a change in the interval between distractors, since such a change could mean that there has been a significant change at the node and it may be necessary to attend to it. In the dataset for this experiment, the interval of presentation began at 0.5 seconds (10 frames), then doubled to 1 second (20 frames) after 10 presentations. Once the distractor was moved 10 times on the 1 second interval, the interval was doubled again to 2 seconds (40 frames). Similarly, after 10 presentations the interval was doubled to 4 seconds (80 frames) for 3 presentations until 990 frames were recorded. Figure 35 shows the nodes in which the distractor moved. Motion occurred in a total of 33 frames. The resulting number of shifts to the distractor for the system with and without habituation are shown in table 15.

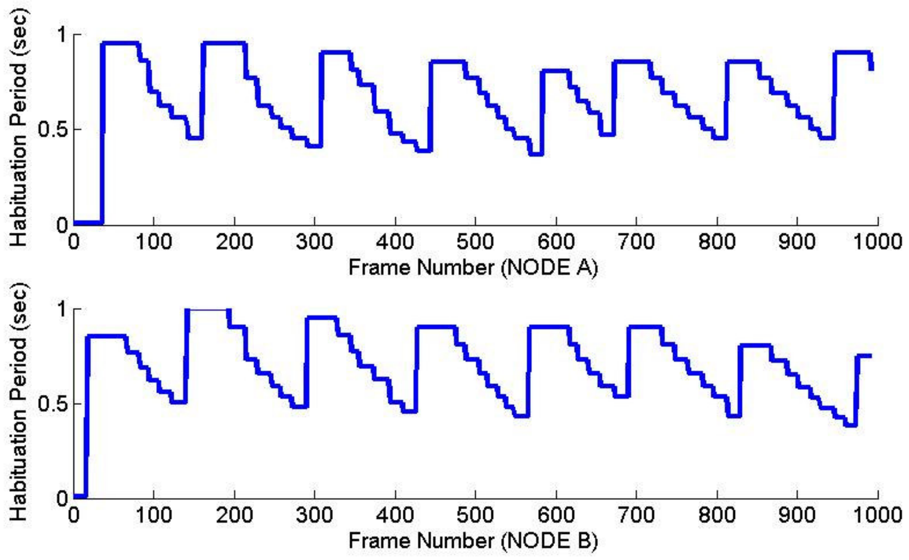
Table 15: Increasing interval motion experiment results

Method	Number (percentages) of frames on distractor
No Habituation	33 (100%)
Habituation (last FOA method)	8 (24.2%)
Habituation (time difference method)	14 (42.4%)

Figure 36 show the FOA shifts to the distractor nodes under both the last FOA and time difference methods. Changes in the periodicity of the distractor occurred at frames 120, 340, and 780, where changes from 0.5 to 1 second (10 to 20 frames),



(a) Last FOA method



(b) Time difference method

Figure 34: *HabituationPeriod* values for distractor nodes for the random interval experiment.

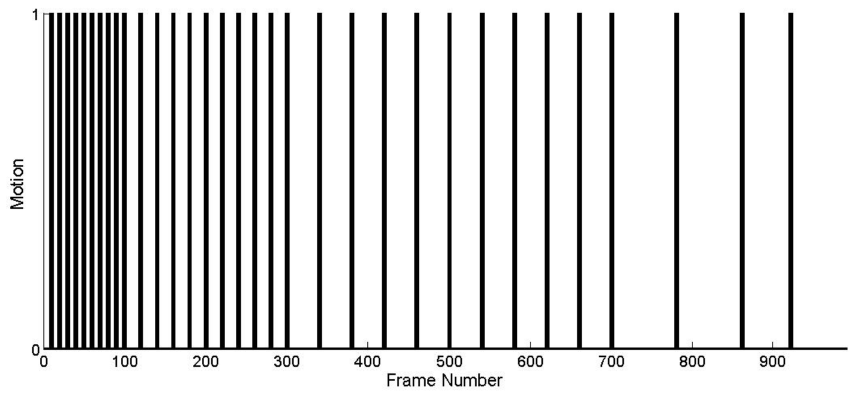
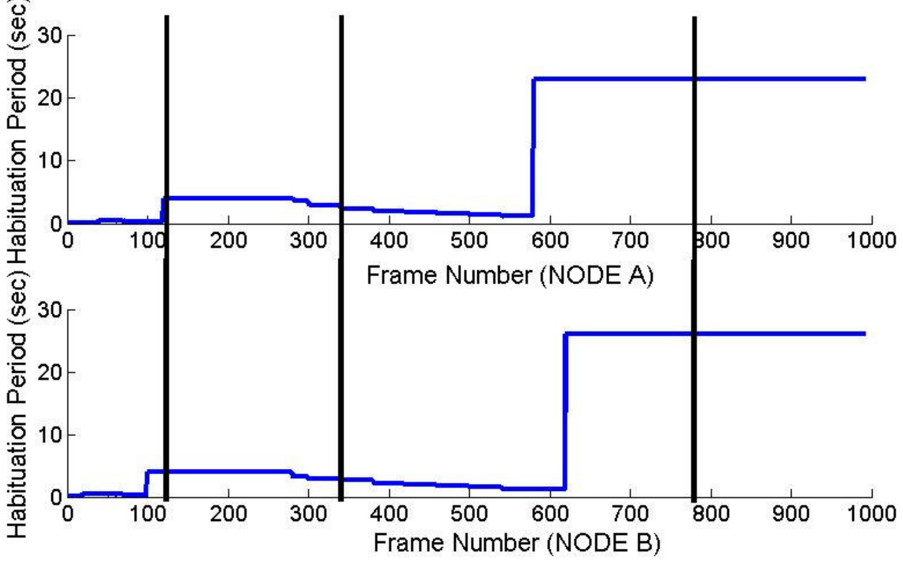


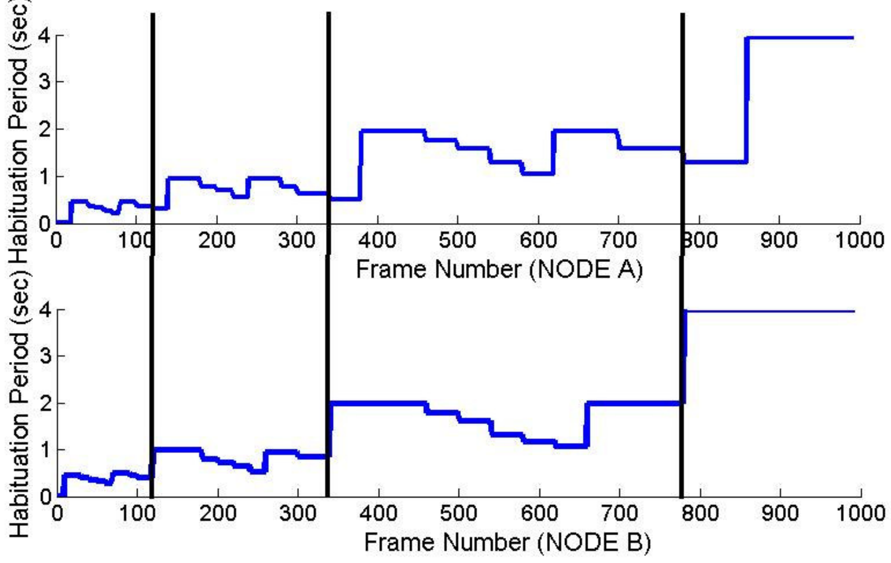
Figure 35: Frames with distracting motion for increasing interval experiment

1 to 2 seconds (20 to 40 frames), and 2 to 4 seconds (40 to 80 frames) took place. These are indicated by the black vertical lines of figure 36. As can be seen, the time difference method is better suited for detecting this type of change. The last FOA method detected the first interval change at frame 120, but this may be due to chance; the *habituationPeriod* variable's decay happened to coincide with the change. It took 240 frames (12 seconds) to detect the 2nd interval change, and the FOA never shifted to the distractor after the interval change to 80 frames before the end of the dataset was reached.

The time difference method yielded better results, as shown in figure 36b. All three interval changes were detected as they occurred (within 0 frames). This is because the maximum value that the *habituationPeriod* can have when the interval is changed corresponds to half of the interval, which would put the new posting in the normal stage of operation and the data would not be habituated. (It would take more than 0 frames to detect an interval change smaller than twice the interval length since the distractor presentation would occur during the recovery stage.) The habituation period can also be observed to become progressively longer as the interval becomes larger.



(a) Last FOA method



(b) Time difference method

Figure 36: *HabituationPeriod* values for distractor nodes for the increasing interval experiment.

Decreasing Interval Experiment

This experiment is similar to the previous one except that the interval between distractors decreases. The dataset begins with a distractor occurring every 2 seconds (40 frames). Once the distractor has occurred 10 times, the interval is halved to 1 second (20 frames) for another 10 times. Similarly, the interval is then halved to 0.5 seconds (10 frames) for the remainder of the dataset. Figure 37 shows the nodes in which the distractor moved. Motion occurred in a total of 59 frames. This dataset was run through the attention system to observe how quickly it could detect a change in the distractor's periodicity. The number of FOA shifts to the distractor for the system with and without habituation are shown in table 16.

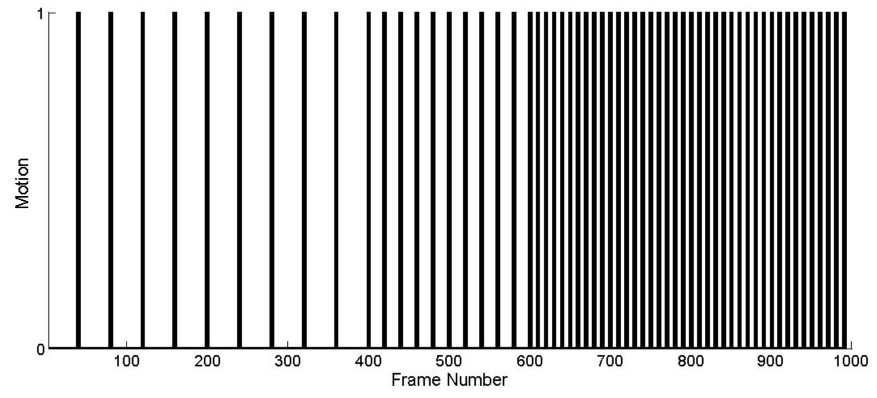


Figure 37: Frames with distracting motion for decreasing interval experiment

Table 16: Decreasing interval motion experiment results

Method	Number (percentages) of frames on distractor
No Habituation	59 (100%)
Habituation (last FOA method)	6 (10.2%)
Habituation (time difference method)	16 (27.1%)

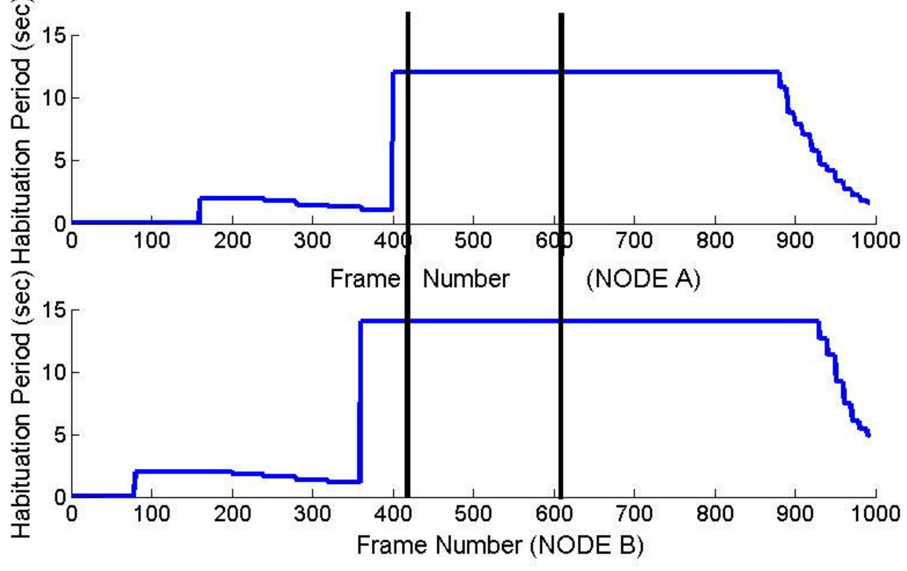
Figure 38 show the FOA shifts to the distractor nodes under both the last FOA and time difference methods. Changes in the periodicity of the distractor occurred at frames 420 and 620, where changes from 2 seconds to 1 second and 1 second to

0.5 seconds took place. These are indicated by the black vertical lines of figure 38. As can be seen, the time difference method is better suited for detecting this type of change. The last FOA method does not detect either of the interval changes before the end of the dataset is reached.

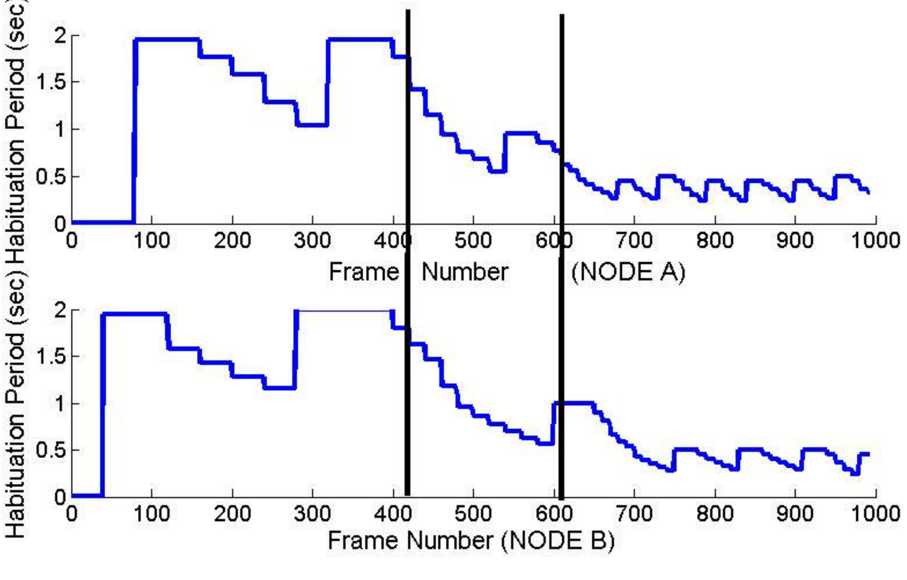
The time difference method yielded better results—shown in figure 38b—although not as good as in the increasing interval experiment. Both interval changes were detected: in this case, the first was detected in 180 frames (9 seconds) and the second in 120 frames (6 seconds). The habituation period can also be observed to become progressively shorter as the interval length decreases.

Depending on when the habituation period is modified and how long it takes for it to decay, an interval change from 2 seconds to 1 second could take up to 360 frames (or 18 seconds at a frame rate of 20fps). This worst case would be the situation where the habituation period has just been reset to 2s immediately prior to the interval change: habituationPeriod would then stay at 2s for 4 seconds (the length of the habituation period plus the recovery period). It would then begin to decay by 10% each time another posting occurs (every 1 second since the interval change has already take place), until the time where it reaches a value approximately half of the current interval length. Similarly for the second interval change, habituation would remain at 1 second for 2 seconds, then decay by 10% every 0.5 seconds (the new interval) until it reaches a value less than half the interval length. This would take a total of 180 frames (or 9 seconds).

The general worst-case time formula for halving an interval can then be expressed as in equation 8. The previous interval is multiplied by two to represent the sum of the habituation and recovery periods. The new interval is multiplied by 14 since it was observed that it takes 14 decays of 10% to reach a value less than half the previous interval length so that a FOA shift can be triggered. This value is multiplied by



(a) Last FOA method



(b) Time difference method

Figure 38: *HabituationPeriod* values for distractor nodes for the increasing interval experiment.

the new interval since the habituation variables are decayed each time a new posting occurs, and these are occurring on the new interval.

$$worstCaseTime = previousInterval * 2 + (14 * newInterval) \tag{8}$$

Demonstration Experiment

This last experiment was performed to demonstrate the overall behavior of the system. A dataset containing 300 frames was recorded. Three tasks having different priorities were active in the locale. The system should attend to the target matching the highest-priority task. A person then walked into the locale, moved the highest-priority target, then walked away. The system should shift its focus of attention to follow the person's movements, then return to the now moving highest-priority target and attend to it. Figure 39 illustrates frames of the dataset at different points in time. The highest priority target is the green one in the center of the image.

The results of the test are shown in table 17. The system focused on the highest-priority (green) target in 87% of the frames. As the person walked through the locale, the FOA shifted from the green target to the person. Once the person had left the locale, the FOA returned to the green target.

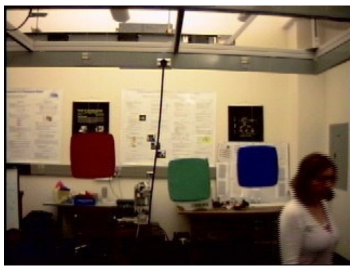
Table 17: Demonstration Experiment Results

Object	Percentages of FOA on object
Highest-Priority Target	(87%)
Person	(13%)

(a) Three stationary targets

(b) Person walks in the locale

(c) Person moves target


(d) Person walks out of the locale

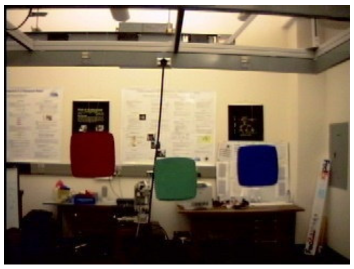

(e) Two stationary targets and one moving target

Figure 39: Selected frames of the dataset

CHAPTER V

CONCLUSIONS AND FUTURE WORK

Conclusions

This dissertation develops a general multimodal sensory interface for guiding a robot's focus of attention (FOA). The system is implemented on a conformal network of locally-connected nodes—the Sensory Ego-Sphere (SES)—and a single FOA is selected on the SES at each time step based on the system's currently defined task. The attention system is preemptive in that it can detect and attend to changes occurring in the robot's locale regardless of the current task. Tasks are ranked in order of priority, and a high-priority task can interrupt a lower-priority task when stimuli matching this new task is detected. Habituation mechanisms are also incorporated into the system to improve performance under the presence of distractors.

Distributed methods for spreading salience-based activation through the local interconnectivity of the nodes are developed. These methods were developed so that a distributed implementation of the SES can be explored in the future. This future implementation would be composed of nodes with independent computing power that can process sensory data in parallel and communicate with their immediate neighbors to set their activation values. Although this functionality is not currently implemented in hardware, the methods of node communication were developed so that they can used on such a distributed system and were explored in simulation in this dissertation.

The system is made up of multiple independent sensory processing modules (SPMs) that post information to the SES at the node closest to the origin of the sensory data. (cf. Chapter III) Any number and type of SPM can be used. These SPMs also assign an activation value to the data during the posting procedure; this activation is calculated based on the current system task, described in terms of the sensory modalities present in the particular system implementation. Another value representing the change in activation is also calculated at each node; this value is used to detect changes in the locale that may match a higher-priority task. Once such a location is identified, the matching higher-priority task becomes the current task and the system attends to the new location. Once the situation resolves and no locations match the higher-priority task, the original task is restored and the system resumes attending to locations that match the original task. Although some changes detected in the system will match higher-priority tasks, others will not and the system should not be distracted by such locations. Habituation mechanisms were incorporated in the system to prevent the FOA from shifting to these distractor locations repeatedly.

Chapter IV describes the experiments that were carried out to validate this attention system. The performance of the system was examined under a variety of conditions. Some experiments were performed with simulated data once it had been shown that the system behaved similarly under real data and simulated data. Care was taken to create simulated datasets that exhibited the same features and noise characteristics observed in real data.

The first experiment identified the best parameter values to use for a few thresholds used by the system to detect a change in the locale that must be attended or a situation matching a high-priority task that has resolved. Once these parameters were identified, an experiment was performed to observe the effects of increasing the number of distractors (chosen to appear similar to the target) in the locale. It was found that system performance is not affected by an increasing number of distractors.

Another experiment was performed to observe the effects of the target's size on system performance. The original target used was approximately seven inches squared (40x40 pixels). The system was tested with targets varying in actual size from seven inches squared down to a one inch square. A target size of 20x20 pixels or larger was found to yield acceptable results. Since processing is performed at each node, the size of the target with respect to the node size is a more significant measurement. Since a target must be approximately 20x20 pixels to be used with the SES tessellation used in this work, the target is approximately half of the node's area. Therefore, one can determine the appropriate SES tessellation for a specific application if the expected target size is known: the SES tessellation should yield nodes no larger than twice the size of the target.

The next experiment was carried out to examine the task-switching behavior of the system, which is dependent on factors such as the initial task chosen at the beginning of the trial, the priority of each task defined in the system as well as its associated sensory data, and whether multiple tasks are allowed to have the same priority. All combinations of these factors were tested with a situation involving a red target and a green target. System performance was measured in terms of percentage of correct FOAs and percentage of correct task selections. It was determined that the order in which equal priority tasks are listed in the priority-ordered task list does not affect performance. The initial task selection, however, does affect system performance under the equal priority condition: whichever task is chosen as the initial task will remain active for the entirety of the experiment. Initial task selection does not affect performance when the tasks are given different priorities; the task with the highest priority will be selected for the entirety of the experiment. When a distractor is moving in the locale, however, the focus of attention is distracted away from the higher-priority target approximately 20% of the time.

Once the general task-switching behavior of the system had been observed, another experiment was performed to determine whether or not increasing the number of possible tasks affected the system performance. The system was tested with the total number of tasks defined in the system ranging from 2 and 6; each task having a matching target in the locale. The targets entered and exited the locale in the same order, and the targets' priorities were alternated in each trial. It was found that although the entry order of the highest-priority task does not affect system performance, the number of tasks/targets in the locale does slightly affect it. Enabling the habituation mechanism does not significantly improve results because the stimuli that causes FOA shifts moves across a large portion of the image (as opposed to over a smaller set of nodes), the motion is not constant or repeated on an interval, and, most importantly, the habituation variables are reset after each task switch, which occurs each time a target enters or exists the locale.

The remainder of the experiments were performed to observe system behavior when habituation mechanisms are enabled in the system. A simple experiment was designed with one target and one periodically moving distractor. Two schemes or applying habituation were investigated: one that used the time between consecutive selections of a node as the FOA to modify habituation variables (last FOA method) and one that used the time between consecutive identical postings to modify the variables (time difference method). Both methods yielded significant improvements over having no habituation mechanism at all. The last FOA method generated the least FOA shifts to distractor nodes, and the time between shifts became progressively longer (which caused the results to improve over time). The time difference method results were better than having no habituation at all in the system, but did not generate as great an improvement as the last FOA method did. The time difference method would be suitable for applications where a distractor must be monitored on a regular interval, yet not as much as would be the case without any habituation. This applies to the situation where habituation variables are allowed to decay back to their initial values over time. If these variables do not decay, the two methods are nearly identical and the distractor is only attended once or twice at the beginning of the experiment.

To examine the effects of the periodicity of the distractor on system performance, several datasets were created: one with a constantly moving distractor, another with a distractor moving on a fixed interval, and yet another with a distractor moving on a randomly changing interval. System performance was measured in terms of the number of FOA shifts to distractor nodes. While both methods of applying habituation yielded better results than having no habituation mechanism, the last FOA method outperformed the time difference method in all cases.

The last two datasets were of a distractor moving on an interval that doubles in size, and another of a distractor that moves on an interval that becomes half its size after some time spent on a fixed interval. Performance of the system here was measured in terms of the time it takes for the system to detect this change. In the case of the increasing interval, although both methods detect the change, the time difference method detects it much faster (at the instant of occurrence). In the case of the decreasing interval, only the time difference method was able to detect the change; the last FOA method did not detect it before the end of the dataset was reached. The maximum time (worst case) needed to detect the change was calculated for the time difference method and found to be dependent on the previous and current time between postings as well as the time since the habituation variables were last set prior to the interval change. An equation was derived to calculate the worst case time using these values. Therefore, if a distractor must only be monitored once in a while, and the time between these FOA shifts can increase with time, the last FOA method should be used. If, however, it is critical for an application to be able to detect changes in the periodicity of a distractor, or to regularly monitor changes in the environment, the time difference method of applying habituation would be more suitable.

A final demonstration dataset was created to observe the overall capabilities of the system. Three targets matching tasks with different priority levels were active in the locale. A person then walks across the locale, pausing to set the target in motion. The system focused on the highest-priority target until the person was detected; the FOA then shifted to the person and followed the person's movements across the locale. Once the person was out of view, the FOA shifted back to the highest-priority target which was then moving. The FOA remained on this target for the remainder of the experiment.

This system was developed with a robot operating in a dynamic environment in mind. This type of robot would need to perform a variety of tasks as well as detect and react to unexpected events while ignoring spurious ones. The developed system is not dependent on the type of sensory modalities used, or on the number of sensory processing modules. Raw data from the environment can be obtained, processed (on a low-level), labeled, and organized by the system and attention can be directed either to the most salient location in the locale or to the location with the most change. Habituation inhibits the change signal so that a distracting location is not repeatedly visited.

Future Work

As stated previously, this system is the first step between raw data and cognitive processing. An evident extension would be to send the information at the node chosen by the FOA selector to higher-level processing modules with more intelligence for further processing and identification. The habituation variables at the FOA node could be modified again based on this second level of processing. If objects can be recognized, the interconnectivity of nodes as well as a labeling scheme previously researched could be leveraged to deploy both attention and habituation more cohesively to distinct objects as opposed to indiscriminately deploying it to all adjacent nodes. This may refine results and improve habituation performance by only attending to an object once if that object extends across multiple nodes, not all of these being adjacent.

All experiments were performed with a stationary robot. This system is also applicable to an SES in motion, but the node positions would need to be shifted with the motion, as described in the appendix of [1].

We would like to leverage the SES's locally-connected structure to explore a parallel implementation of the SES. In our implementation, local processing and interactions modulate a global focus-of-attention selector. If each node were given independent computing power, the ability to aggregate and associate sensory information, as well as the ability to directly communicate with neighboring nodes, the distributed approach could provide faster sensory processing for better real-time behavior.

Depending on the application that the system is used for, different schemes of modifying habituation could be developed. Since we have identified two different methods of applying habituation, the method could be tied to the task. There are perhaps some tasks where stimuli must only be attended once or twice, and the last FOA method could be used in these cases. If, however, a task is sensitive to the periodicity of the stimuli or the stimuli must be monitored regularly, the time difference method of applying habituation could be used when such a task is currently active. It may also be advantageous not to reset all habituation variables after all task changes, especially since these variables decay back to their initial values over time.

REFERENCES

[1] R. A. Peters II, K. A. Hambuchen, and R. Bodenheimer, "The sensory ego-sphere: A mediating interface between sensors and cognition," *Autonomous Robots*, vol. 26, pp. 1–29, January 2009.

[2] K. Achim, "Image mapping and visual attention on a sensory ego-sphere," Master's thesis, Vanderbilt University, August 2005.

[3] J. Wolfe and T. Horowitz, "What attributes guide the deployment of visual attention and how do they do it?," *Nature Reviews Neuroscience*, vol. 5, pp. 495–501, June 2004.

[4] M. I. Posner and S. E. Petersen, "The attention system of the human brain," *Annual Review of Neuroscience*, vol. 12, pp. 25–42, March 1990.

[5] G. R. J. Christoffersen, "Habituation: Events in the history of its characterization and linkage to synaptic depression. a new proposed kinetic criterion for its identification," *Progress in Neurobiology*, vol. 53, pp. 45–66, September 1997.

[6] E. L. Hall, *Handbook of Industrial Automation.* CRC Press, 2000.

[7] iRobot, "Roomba." http://store.irobot.com/.

[8] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, pp. 56–59, April 1965.

[9] B. Hendy, "Pixels per dollar based on australian dollar recommended retail price of kodak digital cameras." Australian PMA DIMA, 1998.

[10] K. Hambuchen, *Multi-modal attention and event binding in humanoid robots using a sensory ego-sphere.* PhD thesis, Vanderbilt University, May 2004.

[11] D. Marr, "Simple memory: a theory for archicortex," *Philosophical Transactions of the Royal Society of London, B*, vol. 262, pp. 23–81, 1971.

[12] I. Stewart, "Circularly covering clatharin," *Nature*, vol. 351, p. 103, 1991.

[13] K. Hambuchen, W. Bluethmann, S. Goza, R. Ambrose, K. Rabe, and M. Allan, "Supervising remote humanoids across intermediate time delay," in *Proceedings of the 2006 IEEE-RAS International Conference on Humanoid Robots*, December 2006.

[14] C. Johnson, A. Koku, K. Kawamura, and R. P. II, "Enhancing a human-robot interface using sensory egosphere," in *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, May 2002.

[15] K. Kawamura, A. B. Koku, D. M. Wilkes, R. A. Peters II, and A. Sekmen, "Toward egocentric navigation," *International journal of robotics and automation*, vol. 17, pp. 135–145, October 2002.

[16] A. L. Edsinger, *Robot Manipulation in Human Environments.* PhD thesis, Massachusetts Institute of Technology (MIT), 2007.

[17] K. A. Fleming, R. A. Peters II, and R. E. Bodenheimer, "Image mapping and visual attention on a sensory ego-sphere," in *Conference on Intelligent Robotics and Systems (IROS)*, October 2006, unpublished.

[18] R. Desimone and J. Duncan, "Neural mechanisms of selective visual attention," *Annual Review of Neuroscience*, vol. 18, pp. 193–222, 1995.

[19] W. James, *Principles of Psychology.* 1890. available online at: http://psychclassics.yorku.ca/James/Principles/index.htm.

[20] M. Posner and M. Raichle, *Images of Mind*, vol. 96. Scientific American Books, 1994.

[21] J. Brefczynski and E. DeYoe, "A physiological correlate of the spotlight of visual attention.," *Nature Neuroscience*, vol. 2, pp. 370–374, 1999.

[22] A. Treisman and G. Gelade, "A feature-integration theory of attention," *Cognitive Psychology*, vol. 12, pp. 87–136, 1980.

[23] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 1254–1259, Nov 1998.

[24] K. Cave, "The featuregate model of visual selection," *Psychological Research*, vol. 62, no. 2, pp. 182–194, 1999.

[25] J. Wolfe, K. Cave, and S. Franzel, "Guided search: An alternative to the feature integration model for visual search," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 15, no. 3, pp. 419–433, 1989.

[26] C. Koch and S. Ullman, "Shifts in selective visual attention: towards the underlying neural circuitry," *Human Neurobiology*, vol. 4, pp. 219–227, 1985.

[27] V. Navalpakkam and L. Itti, "Modeling the influence of task on attention," *Vision Research*, vol. 45, pp. 205–231, Jan 2005.

[28] D. Walther, L. Itti, M. Riesenhuber, T. Poggio, and C. Koch, "Attentional selection for object recognition - a gentle way," in *Lecture Notes in Computer Science*, vol. 2525, pp. 472–479, Nov 2002.

[29] J. A. Driscoll, R. A. Peters II, and K. R. Cave, "A visual attention network for a humanoid robot," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998.

[30] J. Wolfe, "Guided search 2.0: A revised model of visual search," *Psychonomic Bulletin & Review*, vol. 1, no. 2, pp. 202–238, 1994.

[31] J. Wolfe and G. Gancarz, "Guided search 3.0 basic and clinical applications of vision science." Dordrecht, Netherlands: Kluwer Academic. 189-192, 1996.

[32] J. Wolfe, "Guided search 4.0: A guided search model that does not require memory for rejected distractors [abstract]," *Journal of Vision*, vol. 1, no. 3, p. 349a, 2001.

[33] C. Breazeal and B. Scassellati, "A context-dependent attention system for a social robot," in *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pp. 1146–1153, August 1999.

[34] A. Ude, V. W. adn L. Li-Heng, and G. Cheng, "Distributed visual attention on a humanoid robot," in *IEEE-RAS Intl' Conference on Humanoid Robots*, pp. 381–386, December 2006.

[35] A. Ude, J. Moren, and G. Cheng, *Humanoid Robots: Human-like Machines*, ch. Visual Attention and Distributed Processing of Visual Information for the Control of Humanoid Robots, pp. 423–436. I-Tech Education and Publishing, 2007.

[36] A. Maki, P. Nordlund, and J.-O. Eklundh, "Attentional scene segmentation: integrating depth and motion," *Computer Vision and Image Understanding*, vol. 78, no. 3, pp. 351–373, 2000.

[37] S. Frintrop, E. Rome, A. Nchter, and H. Surmann, "A bimodal laser-based attention system," *Computer Vision and Image Understanding*, vol. 100, no. 1-2, pp. 124–151, 2005.

[38] G. Heidemann, "Focus-of-attention from local color symmetries," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 817– 830, July 2004.

[39] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, pp. 147–151, 1988.

[40] Y.-F. Ma and H.-J. Zhang, "Contrast-based image attention analysis by using fuzzy growing," in *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pp. 374–381, November 2003.

[41] K. Lee, H. Buxton, and J. Feng, "Cue-guided search: a computational model of selective attention," *IEEE Transactions on Neural Networks*, vol. 16, no. 4, pp. 910– 924, 2005.

[42] G. Backer and B. Mertsching, "Two selection stages provide efficient object-based attentional control for dynamic vision," in *Proceedings of the International Workshop on Attention and Performance in Computer Vision*, p. 916, 2003.

[43] M. T. Lpez, A. Fernndez-Caballero, M. A. Fernndez, J. Mira, and A. E. Delgado, “Dynamic visual attention model in image sequences,” *Image and Vision Computing*, vol. 25, p. 597613, 2007.

[44] P. Locher and C. Nodine, *Eye Movements: From Physiology to Cognition*, ch. Symmetry catches the eye, pp. 353–361. B.V. (North Holland): Elsevier Science, 1987.

[45] K. Nakayama and G. H. Silverman, “Serial and parallel processing of visual feature conjunctions,” *Nature*, vol. 320, pp. 264–265, 1986.

[46] G. Deco, “Biased competition mechanisms for visual attention in a multimodular neurodynamical system,” in *Emergent Neural Computational Architectures Based on Neuroscience : Towards Neuroscience-Inspired Computing*, Lecture Notes in Computer Science, pp. 114–126, Springer Berlin / Heidelberg, 2001.

[47] M. Begum, G. Mann, and R. Gosine, “A biologically inspired bayesian model of visual attention for humanoid robots,” in *IEEE-RAS Intl' Conference on Humanoid Robots*, pp. 587–592, December 2006.

[48] M. Jagersand, “Saliency maps and attention selection in scale and spatial coordinates: an information theoretic appreach,” in *Proceedings of the 5th International Conference in Computer Vision*, pp. 195–202, June 1995.

[49] J. K. Tsotsos, S. M. Culhane, W. Y. K. Wai, Y. Lai, N. Davis, and F. Nuflo, “Modeling visual attention via selective tuning,” *Artificial Intelligence*, vol. 78, no. 1-2, pp. 507–545, 1995.

[50] A. L. Rothenstein and J. K. Tsotsos, “Attention links sensing to recognition,” *Image and Vision Computing*, vol. 26, no. 1, pp. 114–126, 2008.

[51] S. Culhane and J. Tsotsos, “A prototype for data-driven visual attention,” in *Proceedings og the International Conference on Pattern Recognition*, pp. 36 – 40, 1992.

[52] D. Heinke and G. W. Humphreys, “Saim: A model of visual attention and neglect,” in *ICANN '97: Proceedings of the 7th International Conference on Artificial Neural Networks*, pp. 913–918, Springer-Verlag, 1997.

[53] M. Aziz, B. Mertsching, M. Salah, E.-N. Shafik, and R. Stemmer, “Evaluation of visual attention models for robots,” in *IEEE International Conference on Computer Vision Systems*, pp. 20–25, Jan. 2006.

[54] A. Haasch, N. Hofemann, J. Fritsch, and G. Sagerer, “A multi-modal object attention system for a mobile robot,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2712– 2717, August 2005.

[55] T. Wilhelm, H. J. Bhme, and H. M. Gross, “A multi-modal system for tracking and analyzing faces on a mobile robot,” *Robotics and Autonomous Systems*, vol. 48, no. 1, pp. 31–40, 2004.

[56] O. Dniz, M. Castrillon, J. Lorenzo, M. Hernndez, and J. Mndez, "Multimodal attention system for an interactive robot," *Pattern Recognition and Image Analysis*, vol. 2652, pp. 212–220, 2003.

[57] S. Lang, M. Kleinehagenbrock, S. Hohenner, J. Fritsch, G. A. Fink, and G. Sagerer, "Providing the basis for human-robot-interaction: a multi-modal attention system for a mobile robot," in *Proceedings of the 5th International Conference on Multimodal Interfaces*, pp. 28 – 35, 2003.

[58] N. Ouerhani and H. Hugli, "Computing visual attention from scene depth," in *Proceedings of the International Conference on Pattern Recognition*, vol. 1, pp. 375–378, September 2000.

[59] R. Milanese, *Detecting Salient Regions in an Image: from Biological Evidence to Computer Implementation.* PhD thesis, University of Geneva, December 1993.

[60] A. Koene, J. Morn, V. Trifa, and G. Cheng, "Gaze shift reflex in a humanoid active vision system," in *Proceedings of the 5th International Conference on Computer Vision Systems (ICVS)*, 2007.

[61] J. L. Crespo, A. Faia, and R. J. Duro, "An implementation of a general purpose attentional mechanism for artificial organisms," in *Nature Inspired Problem-Solving Methods in Knowledge Engineering*, Lecture Notes in Computer Science, pp. 99–108, Springer Berlin / Heidelberg, 2007.

[62] L. Goncalves, "Towards a learning model for feature integration in attention control," in *International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 311– 316, 2001.

[63] S. Rajendran and M. Huber, "Learning task-specific sensing, control and memory policies," *International Journal on Artificial Intelligence Tools*, vol. 14, no. 1-2, pp. 303–327, 2005.

[64] S. Baluja and D. Pomerleau, "Dynamic relevance: Vision-based focus of attention using artificial neural networks," in *Artificial Intelligence 87*, pp. 381–395, 1997.

[65] F. W. M. Stentiford, "An attention based similarity measure with application to content-based information retrieval," in *Proceedings of the Storage and Retrieval for Media Databases conference, SPIE Electronic Imaging*, vol. 5021, January 2003.

[66] A. Bamidele, F. W. M. Stentiford, F. W. M. Stentiford, and J. Morphett, "An attention-based approach to content-based image retrieval," *BT Technology Journal*, vol. 22, no. 3, pp. 151–160, 2004.

[67] C. Breazeal, A. Edsinger, P. Fitzpatrick, and B. Scassellati, "Active vision for sociable robots," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 31, pp. 443–453, September 2001.

[68] T. Rogers, A. Sekmen, and J. Peng, "Attention mechanisms for social engagements of robots with multiple people," in *Intl' Symposium on Robot and Human Interaction communication (ROMAN)*, pp. 605–610, September 2006.

[69] M. Doniec, S. Ganghua, and B. Scassellati, "Active learning of joint attention," in *IEEE-RAS Intl' Conference on Humanoid Robots*, pp. 34–39, December 2006.

[70] T. Wada and T.Matsuyama, "Multiobject behavior recognition by event driven selective attention method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 873 – 887, 2000.

[71] F. Shic and B. Scassellati, "A behavioral analysis of computational models of visual attention," *International Journal of Computer Vision*, vol. 72, pp. 159–177, June 2006.

[72] T. Williams and B. Draper, "An evaluation of motion in artificial selective attention," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 3, pp. 85–92, 2005.

[73] N. Ouerhani and H. Hgli, "Real-time visual attention on a massively parallel simd architecture," *Real-Time Imaging*, vol. 9, no. 3, pp. 189–196, 2003.

[74] G. Sela and M. D. Levine, "Real-time attention for robotic vision," *Real-Time Imaging*, vol. 3, no. 3, pp. 173–194, 1997.

[75] A. Sha'ashua and S. Ullman, "Structural saliency: The detection of globally salient structures using a locally connected network," in *International Conference on Computer Vision*, pp. 321–327, December 1988.

[76] M. Schlosser and K. Kroschel, "Dedicated mechanisms for the attention system in humanoid robots," in *IEEE-RAS Intl' Conference on Humanoid Robots*, pp. 52–63, November 2004.

[77] C. Balkenius, "Attention, habituation and conditioning: toward a computational model," *Cognitive Science Quarterly*, vol. 1, no. 2, pp. 171–214, 2000.

[78] H. Pinsker, I. Kupfermann, V. Castellucci, and E. Kandel, "Habituation and dishabituation of the gm-withdrawal reflex in aplysia," *Science*, vol. 167, no. 3926, pp. 1740 – 1742, 1970.

[79] R. L. Fantz, "Visual experience in infants: Decreased attention to familiar patterns relative to novel ones," *Science*, vol. 146, no. 3644, pp. 668 – 670, 1964.

[80] S. Sirois and D. Mareschal, "An interacting systems model of infant habituation," *Journal of Cognitive Neuroscience*, vol. 16, no. 8, pp. 1352–1362, 2004.

[81] R. Thompson and W. Spencer, "Habituation: A model phenomenon for the study of neuronal substrates of behavior," *Psychology Review*, vol. 73, pp. 16–43, 1966.

[82] A. S. Bristol, A. L. Purcell, and T. J. Carew, *The Handbook of Brain Theory and Neural Networks*, ch. Habituation, pp. 504–507. MIT Press, 2nd ed., 2003.

[83] J. Staddon, "Interval timing: memory, not a clock," *Trends in Cognitive Science*, vol. 9, pp. 312–314, July 2005.

[84] J. C. Stanley, "Computer simulation of a model of habituation," *Nature*, vol. 261, pp. 146–148, May 1976.

[85] Q. Meng and M. Lee, *Biomimetic Neural Learning for Intelligent Robots*, ch. Novelty and Habituation: The Driving Forces in Early Stage Learning for Developmental Robotics, pp. 315–332. Springer Berlin / Heidelberg, 2005.

[86] S. Marsland, U. Nehmzow, and J. Shapiro, "Detecting novel features of an environment using habituation," in *From Animals to Animats, Sixth Intl Conference on Simulation of Adaptive Behaviour*, (Paris, France), pp. 189–198, 2000.

[87] D. Wang and M. A. Arbib, "Modeling the dishabituation hierarchy: The role of the primordial hippocampus," *Biological Cybernetics*, vol. 67, pp. 535–544, October 1992.

[88] E. Solokov, *The Central Nervous System and Behavior*, ch. Neuronal models and the orienting reflex, pp. 187–276. 1960.

[89] P. Crook and G. Hayes, "A robot implementation of a biologically inspired method for novelty detection," in *Proceedings of Towards Intelligent Mobile Robots*, (Manchester), April 2001.

[90] M. Peters and A. Sowmya, "Integrated techniques for self-organisation, sampling, habituation, and motion tracking in visual robotics applications," in *Proceedings of IAPR Workshop on Machine Vision Applications*, 1998.

[91] S. Sirois, "Hebbian motor control in a robot-embedded model of habituation," in *IEEE-INNS-ENNS International Joint Conference on Neural Networks*, vol. 5, pp. 2772– 2777, August 2005.

[92] O. Dniz, J. Lorenzo, and M. Hernndez, "A simple habituation mechanism for perceptual user interfaces," *Revista Iberoamericana de Inteligencia Artificial*, vol. VIII, no. 23, pp. 7–16, 2004.

[93] B. Stiles and J. Ghosh, "A habituation based neural network for spatio-temporal classification.," in *Proceedings of the IEEE Workshop in Neural Networks for Signal Processing*, pp. 135–144, September 1995.

[94] C. Chang, "Improving hallway navigation in mobile robots with sensor habituation," in *IEEE-INNS-ENNS International Joint Conference on Neural Networks*, vol. 5, p. 5143, July 2000.

[95] C. Bishop, *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[96] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational properties,” *Proceedings of the National Academy of Sciences*, vol. 79, pp. 2554–2558, 1982.

[97] M. J. E. Golay, “Hexagonal parallel pattern transformations,” *IEEE Transactions on Computers*, vol. C-18, pp. 733–740, Aug 1969.

[98] E. Deutsch, “On parallel operations on hexagonal arrays,” *Transactions on Computers*, vol. C-19, pp. 982–983, October 1970.

[99] J. A. Snyman, *Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms.* Springer, 2005.