# Utilization and viability of biologically-inspired algorithms in a dynamic multi agent camera surveillance system.

T. Nathan Mundhenk[a,c,1], Nitin Dhavale[a], Salvador Marmol[a], Elizabeth Calleja[a], Vidhya Navalpakkam[a], Kirstie Bellman[c] , Chris Landauer[c] , Michael A. Arbib[a? ,b] , Laurent Itti[a,b]

[a]Computer Science Department, Henry Salvatori Computer Science Center, University of Southern California, Los Angeles, CA 90089-0781;
[b]Neuroscience Program, Hedco Neuroscience Building, University of Southern California, Los Angeles, CA 90089-2520;
[c]Aerospace Integration Science Center, Aerospace Corporation, 2350 E. El Segundo Blvd, El Segundo, CA 90245-4691

## ABSTRACT

In view of the growing complexity of computational tasks and their design, we propose that certain interactive systems may be better designed by utilizing computational strategies based on the study of the human brain. Compared with current engineering paradigms, brain theory offers the promise of improved self-organization and adaptation to the current environment, freeing the programmer from having to address those issues in a procedural manner when designing and implementing large-scale complex systems. To advance this hypothesis, we discus a multi-agent surveillance system where 12 agent CPUs each with its own camera, compete and cooperate to monitor a large room. To cope with the overload of image data streaming from 12 cameras, we take inspiration from the primate's visual system, which allows the animal to operate a real-time selection of the few most conspicuous locations in visual input. This is accomplished by having each camera agent utilize the bottom-up, saliency-based visual attention algorithm of Itti and Koch (Vision Research 2000;40(10-12):1489-1506) to scan the scene for objects of interest. Real time operation is achieved using a distributed version that runs on a 16-CPU Beowulf cluster composed of the agent computers. The algorithm guides cameras to track and monitor salient objects based on maps of color, orientation, intensity, and motion. To spread camera view points or create cooperation in monitoring highly salient targets, camera agents bias each other by increasing or decreasing the weight of different feature vectors in other cameras, using mechanisms similar to excitation and suppression that have been documented in electrophysiology, psychophysics and imaging studies of low-level visual processing. In addition, if cameras need to compete for computing resources, allocation of computational time is weighed based upon the history of each camera. A camera agent that has a history of seeing more salient targets is more likely to obtain computational resources. The system demonstrates the viability of biologically inspired systems in a real time tracking. In future work we plan on implementing additional biological mechanisms for cooperative management of both the sensor and processing resources in this system that include top down biasing for target specificity as well as novelty and the activity of the tracked object in relation to sensitive features of the environment.

## 1. INTRODUCTION

### 1.1 The Goal

The goal of this project is to create a multi-agent distributed real time surveillance camera network based upon biologically based neural algorithms. Additionally, another goal is to create a basic flexible facility for experiments with distributed visual agent experimentation. Also, with the need for additional security in today's world, we are deploying our system with certain applications in mind. The idea is such a network can watch or help watch a secure area. For instance, an intelligent network could enable one human operator to effectively monitor large batteries of cameras. In our paradigm, camera agents in the network would choose scenes to bring to a human operator's

[1] mundhenk@usc.edu; University of Southern California, 3641 Watt Way, HNB 06, Los Angeles Ca, 90089-2520

attention. Thus, an application of our system is to prevent information overload where many channels of video information are available. As such we have laid down several goals, which include:

(1) The multi-agent network should be robust – as is common, we would like our multi-agent network to work in a variety of scenes in a variety of situations with as little software adjustment as possible.
(2) The network should approach plug and play applicability – it is important that if our network is applied in the real world, it should deploy without the need for extensive technical knowledge.
(3) The network should have a useful "understanding" of activity in the area it surveys – for instance, it should have an idea of unusual behavior by objects or of what behavior should attract its attention.

### 1.2 The ingredients

Sensing, perception and inference in the world is something humans do quite naturally, and even with limitations that have been discovered with the way in which humans do such things, no other machine on the planet is nearly as well developed to understand and react to the world as the human brain. While it is without a doubt extremely complex, the human brain offers clues to ways in which real world AI applications can be developed and deployed. Further, in addition to the brains ability to solve problems, it is also extremely adept at generalizing and associating where incomplete knowledge may be present or a situation may have not been encountered. For instance, if I drive into Pennsylvania and the asphalt turns red from the local stone, I am able to infer that I am still following a road. A traditional program with rigid parameters would not have such flexibility and as such may choose to dismiss the new red asphalt as being something other than the road. Thus, the burden on the programmer using non-biological mechanisms frequently becomes the need to attempt to define all possible solutions to a problem. As we will show, the biological solution draws its strength from making fewer, more generalizable assumptions about the world.

By utilizing the biological approach we gain at the first level generalizability and associability. However, the process of doing so also involves linking brain operating units together. That is, as algorithms are discovered that describe the workings of different units in the brain, even more strength will be gained by linking them together. Thus, the second task in creating biological solutions we believe involves construction of larger scale schemas of connected schemas. From this, the human brain can be modeled and emulated in more complex manners as parts are understood and constructed. That is, for instance, if a modular schema is created for executive thought processes and one is created for bottom up visual attention, the two can be combined to create a new algorithm, which uses executive functions to bias, a bottom up visual saliency system from the top down. Such a method will be described in greater detail later.

For our task at hand of creating a multi agent camera network, we have the additional task of integrating schemas

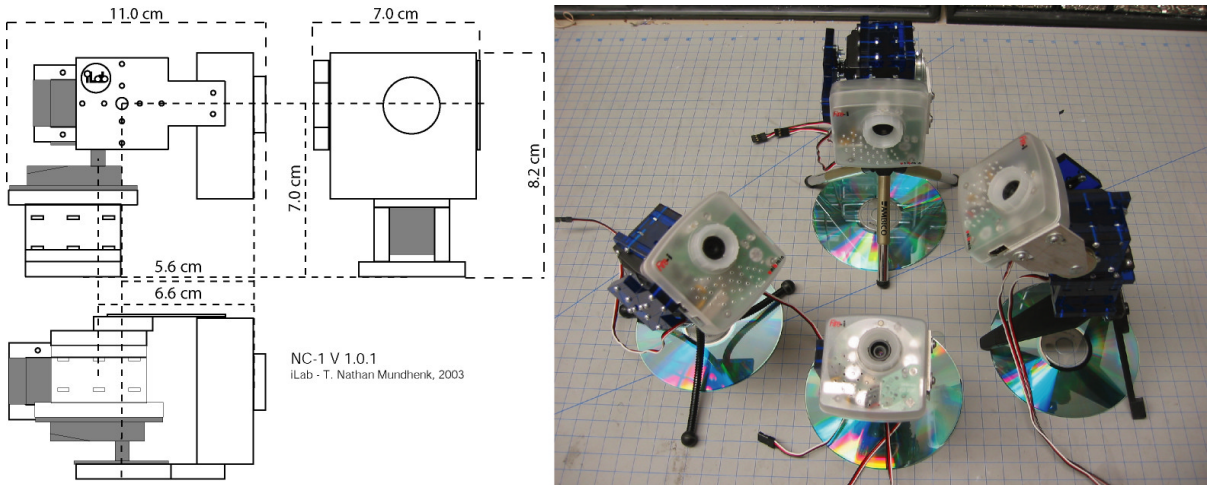


Figure 1: The left image is a basic schematic for the NC-1 two-degree of freedom robotic camera. The right image shows four of the completed cameras with CD-ROMs for scale.

and models not only directly on single machines, but also across machines and a multitude of devices. Thus, we must have a way of integrating the large scale system we have created. As we will discuss later, our approach to this is to modularize and carefully describe each unit in a manner known as wrapping. In this methodology, each module of the distributed system is treated as a resource, which comes with a "wrapping" that, provides a succinct discussion of the module's usability parameters. The aim is to support the automatic integration of separately wrapped components by an agent that has knowledge of a task and can use their wrappings to identify appropriate modules and plan their integration. We have completed such a resource manager for other applications [10,11] but the design of such a system within the present context remains a target for current research. However, our understanding of the need for wrappings has shaped the research described here. The proposed research will ensure that future versions of our large-scale system will be integrated more quickly than if each component were hand integrated, as at present. Additionally, since different resources can perform the same task (for instance both a motor cycle and a car can drive you to work), if one resource is unavailable, system wide wrapping allows an agent to select an appropriate substitution if the primary resource becomes unavailable.

### 1.3 The test bed

We are interested in testing the ability of biological systems to integrate and perform complex real world tasks. As such, we have designed a distributed network of cameras to monitor a room. The eventual goal is to have a ubiquitous camera network that can monitor an area and have a general understanding of the objects it encounters as well as the actions being performed. Additionally, the network must be able to interact or react to knowledge it gains from its observations. As such the biological approach provides many opportunities for solutions in our network. For instance, knowledge from visual attention and salience research has allowed the development of a real time bottom up system for attending to objects in a room. Biology further gives us a model for distributing attention across camera agents by using principles of executive attention competition gleaned from knowledge of executive attention in humans. This creates both the ability to distribute attention as well as bias attention to important events and objects. In complementary research, we will distinguish those objects which are passive in character from "actors" which are autonomously changing in ways unknown to the "brains" of the computer network. Future work in this direction will build on our biological studies of brain mechanisms for visually-directed grasping [9] to understand how the *effectivities* (capacities for action) of observed actors may be related to the *affordances* (potentialities for being acted upon) of observed objects to provide an anticipatory component to the tracking of actors.

## 2. HARDWARE SYSTEMS

The base platform for the camera network is comprised of a 16 single CPU PC Beowulf cluster. Each of the 16 nodes is a standard Pentium III 733 MHz CPU based system. To integrate the Beowulf, the Mandrake Linux operating system is used. This allows not only ease of development for distributed computing ,but it also has built in facilities for network bonding on standard low cost Ethernet components, which is a simple concept whereby a computer has multiple network connections working in parallel. On the Beowulf cluster, three 100Base-T connections working in separate but parallel networks allow networking speeds of up to 600 Mbs in asynchronous mode. At this speed video can be streamed simultaneously between Beowulf nodes.

Video input can be provided by various methods. For the present study, we have created custom two degree of freedom cameras we call the NC-1[2] (see figure 1). Each one consists of two high torque RC servo motors assembled with a Plexiglas frame. We then mount a Unibrain[3] Fire-I firewire based camera to it. The camera is capable of streaming real time digital video in full YUV422 color at a resolution of 640 x 480 pixels. The servomotors are controlled by the Mini-SSC II RC servo controller[4]. This allows control of the camera mount servos from a standard RS-232 serial port. The NC-1 is built with a standard tripod mount and can as such be placed almost anywhere. With the introduction of extended specs for IEEE1394 firewire[5], cables to connect the video input are now available in

[2] A how-to is provided for building this camera at: http://www.nerd-cam.com/how-to/
[3] More info at http://www.unibrain.com/
[4] More info at http://www.seetron.com/ssc.htm
[5] Details on these specs may be found at http://www.unibrain.com/1394_products/cables/cables.htm

lengths as long as 20 meters. In addition, using a repeater can extend the length of a firewire cable to 72 meters. Thus, the NC-1 can be placed anywhere in an area so long as it is within 72 meters of the PC it is connected to.

## 3. SOFTWARE SYSTEMS

### 3.1 Visual Attention and Saliency with the iLab Neuromorphic Toolkit[6]

The first component to our system is the visual saliency model first introduced by Itti and Koch [2,3]. This model, which can be seen in figure 2, is based upon the structure of early visual attention mechanisms in the human brain. The algorithm works by analyzing visual features known to be important in human visual attention such as orientation of objects in a scene, luminance of objects, color opponency, relative motion of objects and the flicker of objects. Attention is focused by allowing features to compete. This means that if two visual items are of similar orientation and proximal, they will tend to cancel each other out. This leaves more unique features intact. The different features are summed together, which creates a saliency map. The most salient point in an image is the most salient point in the combined saliency map. The computer then attends to this winner in a winner take all paradigm.

To date tests have shown that the saliency algorithm is highly effective at locating salient objects without an imposed bias for what it feels it should look for. Thus, it is effective at picking out odd man out objects [2,3], Tanks in a natural scene [4] and other objects in natural scenes. Further, the saliency algorithm works in a manner very similar to human bottom up attention as has been demonstrated by comparing the results to the eye movement of volunteers exposed to the same visual stimuli in our lab.

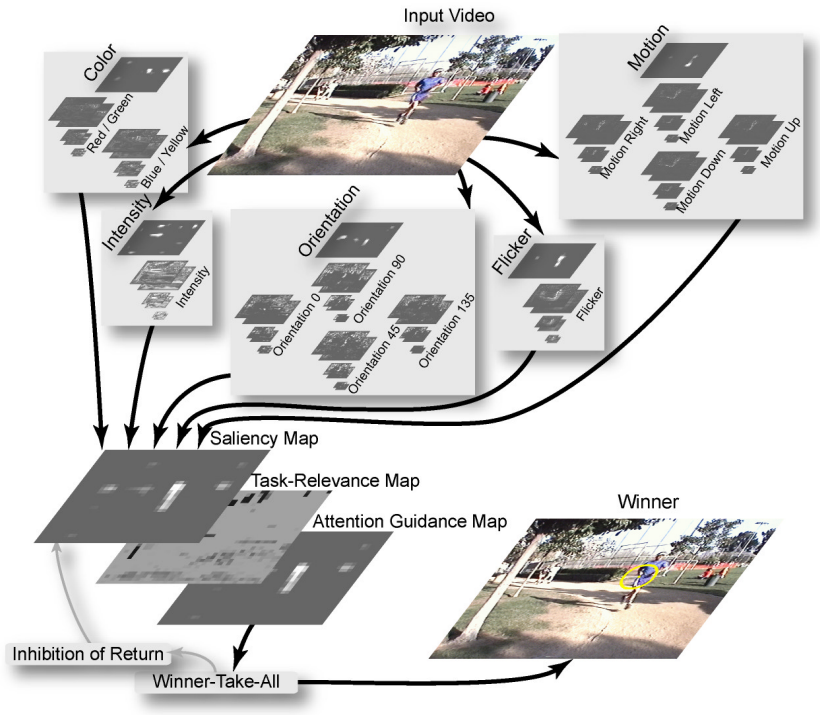


Figure 2: Visual Saliency works in real time by taking in a video stream. Each frame is processed to extract information such as intensity, color orientation, flicker and motion. The activity map in each channel then competes to boost more strong or unique output in channels. The resulting maps are brought together into a combined saliency map. The most salient object is select by a winner take all paradigm. This object can then be attended to.

[6] More info at: http://ilab.usc.edu

### 3.2 Brain Operating Principles and the Neural Simulation Language

The visual saliency algorithm described fulfills the task of bottom up attention. This allows single cameras to attend to some object in a scene. However, this alone does not make an autonomous surveillance system, especially if that system includes multiple agent cameras. If we have 12 cameras in an area and one person walks in wearing a very bright red shirt, all cameras might decide to attend to that person. Thus, with only a bottom up attention paradigm, attention gets no guarantee of reasonable distribution. Additionally, a bottom up attention paradigm is "stateless" in the sense that it does not have a "state" which summarizes context and relevant history to provide an appropriate bias to attention. For instance, in a secured area where all people wear a standard uniform. Whatever the salience of that uniform, if a person comes into the scene without expected attire, say in a T-shirt and jeans, then this should be given a higher salience. Thus, for this example, context drives a bias for saliency. For humans there are many models about the maintenance of contextual information in working memory [7] as well as working memory for a current task [5,6]. One hypothesis on the working of these models in the human brain is that scene information is extracted and filtered into an abstract semantic representation of the scene.

In our lab's effort, we are afforded some of these abilities by extracting features of an object from the visual saliency map from the saliency algorithm. The manner in which this works is at its core very simple. When the saliency algorithm attends to an object, we can extract the features – luminance, color orientations, etc. - at the attended location. Depending on the scale of the various feature maps, each sample taken in this process summarizes feature values across neighborhoods of size 4x4 to 256x256. That is, we know its luminance, colors, orientations etc. at that location. These features give a general idea of the properties of an object. Thus, an elementary object categorization can be implemented by sorting feature vectors into a feature space and classifying new objects by finding out where they fall in that feature space. Once we know the properties of objects in an area, we can bias the saliency algorithm to have a higher probability of spotting objects we know from history to be more interesting. From this we can run our system from the context afforded from an executive. Given this overview, we will now describe these things in greater detail.

#### 3.2.1 The Neural Simulation Language[7]

In order to implement neural like systems we use the Neural Simulation Language (NSL) [1]. This is an intermediate level neural simulation language that allows to easy construction of neural modules and their connection. Once an operating mechanism for the brain has been implemented in NSL, it can be linked to other modules to create an even larger and more complete machine. For instance, a module which selects the maximum stimulus [1] can be combined with one that produces memory-matching outputs. This allows the new NSL program to select the best matching memory. Additionally, once this module is complete, other modules can be built upon it.

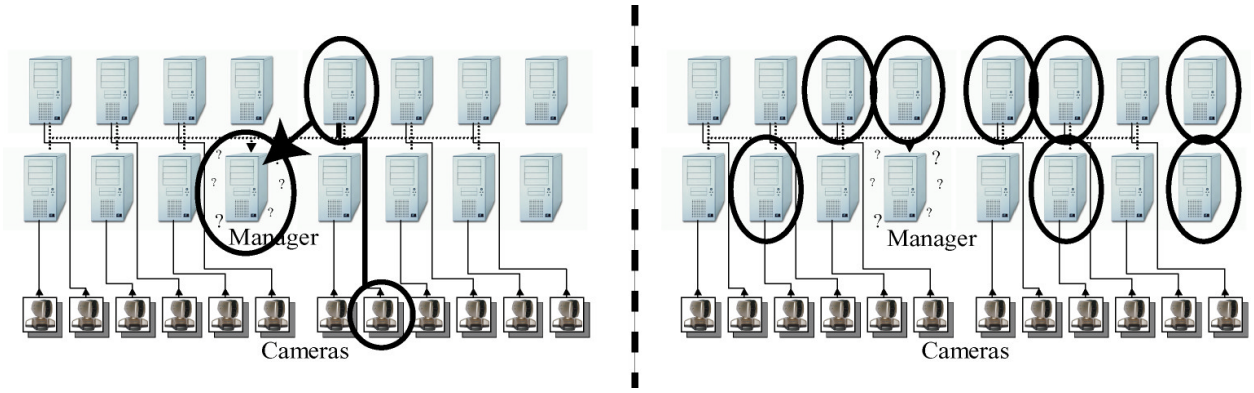


Figure 3: In the left frame, a camera takes an image and requests from the manager access to the saliency program in the Beowulf cluster. In the right frame the manager has determined the Beowulf load and granted access for the agent camera to have access to eight computers, which will process the saliency, of the image it has. When completed, the agent computer will saccade to the most salient location in its visual field.

[7] More info at: http://nsl.usc.edu or at http://www.neuralsimulationlanguage.org/

For the surveillance system, NSL will be used to balance competing interests by agent cameras in a neurophysiological manner. Each agent will have its own NSL based brain connected via distributed NSL which under development. NSL will allow the system to link functional neural modules as needed. This will be extended by wrappings as discussed below, which will make this linking more effortless. Section 3.2.3 talks in greater detail about the distributed attention network.

### 3.2.2 Assembling an agent network

The multi-agent network will need to distribute tasks as well as cooperate on some. For instance, the nodes that are camera agents will have to work as saliency nodes. That is, the visual saliency program that runs in real time across the Beowulf cluster will work in parallel on the same systems that are the agent camera computers. The saliency program uses 8 nodes of the 16 in the Beowulf cluster meaning that two saliency processes can run in parallel. Each camera agent will need to request resources from the Beowulf. For the most part, individual camera agents should always have their requests fulfilled. This is because a camera agent will only make a request in the current implementation between saccades (camera movement). Additionally, the saliency program only loads each CPU at 10%. This means most of the CPU's resources will be available for other agent tasks. With a minimum performance of 30 frames per second on two Beowulf sub-clusters, the Beowulf will process 60 frames per second. At the very least, each of the 12 camera agents will have 5 saccade requests grantable per second. However, we believe that it is possible that requests may be denied if circumstances become less ideal. As such, load balancing for saliency requests will be handled by a manager, which will grant permission to each agent camera to use the saliency program. The scheduling will be controlled by several factors. First, camera agents with a history of seeing more interesting things will have a greater priority. However, this priority will be adjusted based upon wait. That is, cameras will not be made to wait an inordinate amount of time even if they have a history of low saliency.

This load balancing like the distributed attention system in 3.2.3 will be handled by NSL. This will also allow integration of camera memory into a central executive that will create a global view of the area being surveyed. That is, while cameras will be responsible for many low level attention activities, some things such as scene understanding would be best done by a central agent that can collect and process information from individual agents. This is due to the fact that like in the human brain, while some processes are very cooperative, some processes require executive oversight such as top-down attention and planning which in the human brain are the realm of executive centers, for instance, the prefrontal cortex.

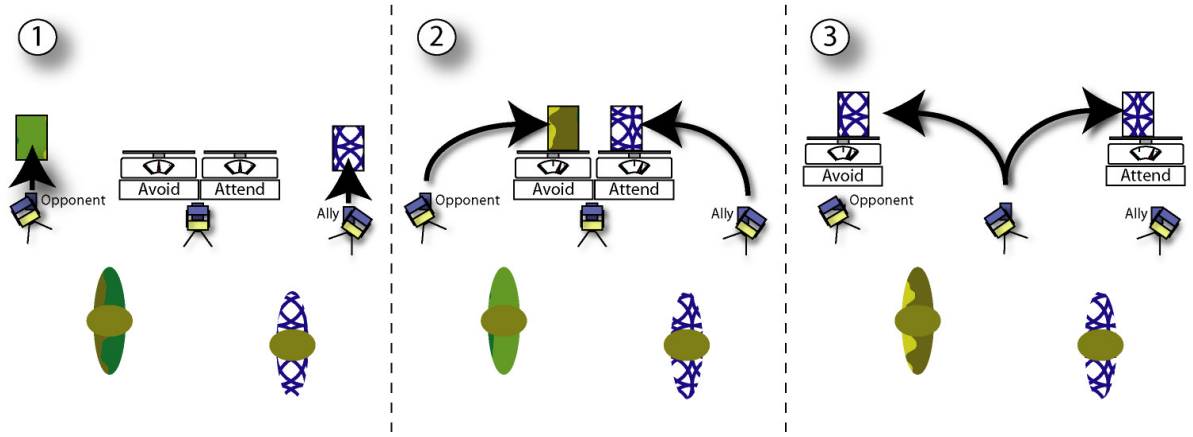


Figure 4: Overhead view: (1) Two cameras attend to two different people and extract the features of those people. (2) The center camera is biased by its ally on the right to attend to the same person, while the opponent bases it away from what it is attending to. (3) Reciprocally, the center camera will bias the other cameras in the same way. This synergy should help create a locking ability between allies.

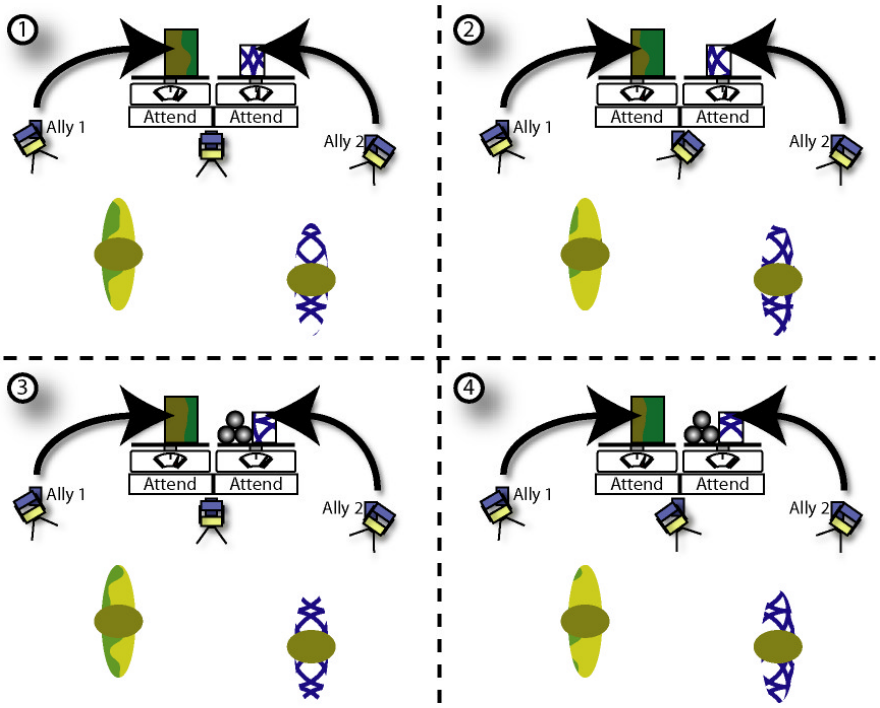


Figure 5: (1) two cameras compete to bias a second cameras attention. (2) Ally 1 wins since its stimulus is more salient. (3) The scenario is repeated, however, ally 2's inputs are given more weight since its view is less similar. (4) Ally 2 wins this time even though its stimulus is less salient.

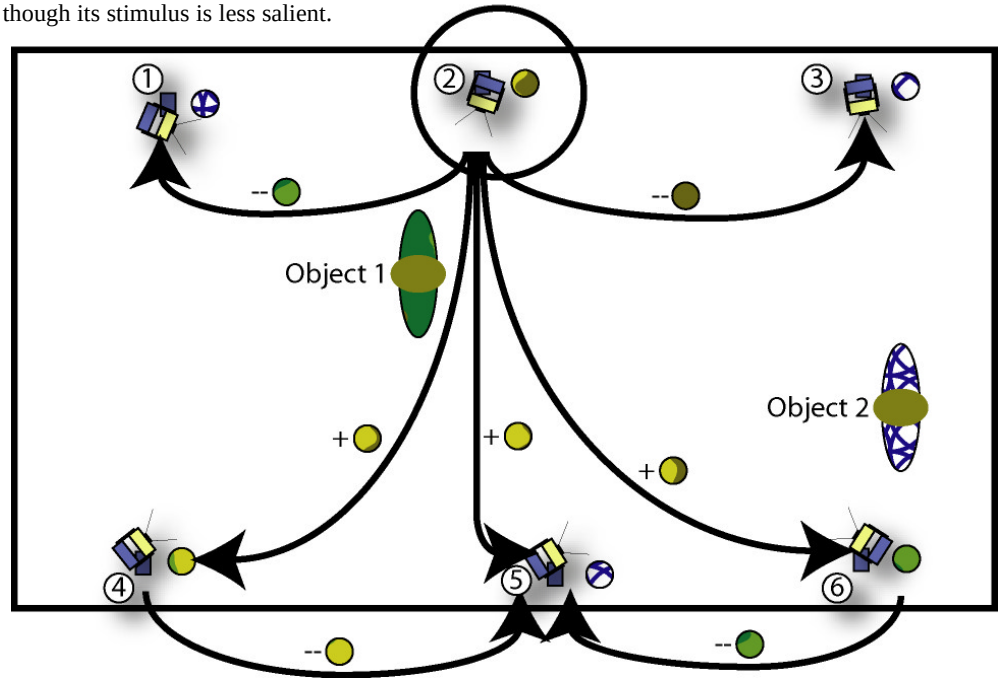


Figure 6: Camera 2 while observing object 1 biases cameras 1 and 3 with similar view points to look at an object with different feature vectors, while camera 4,5 and 6 with dissimilar view points are biased to look at a the same type of object. However, Cameras 4 and 6 bias camera 5 away from their observed stimulus. Thus, in this scenario, attention is completely distributed in an alternating fashion. This can thought of like other self-organizing systems such as the ones that create alternating stripes in certain fish [8].

### 3.2.3 Dividing attention from task based working memory

The agent cameras should desire to divide attention when an area is filled with less interesting things, or alternatively to fixate more camera agents onto an object when it has determined that it is very interesting. The first step in doing this is understanding the capability of other agent cameras. When you know, for instance, the location of other cameras in an area, you can attempt to bias cameras with similar view points away from your own, while attempting to recruit cameras with dissimilar view points to help you watch an object. The reason for doing such a thing is to give an operator many views of an object and avoid too much overlap. Thus, we would like to see a suspicious person from different angles more often than from the same angles. This can be accomplished by exercising principles that the brain uses for attention, but at a more executive level. Figure 4 shows an example of camera agents biasing each other's attention.

Attention is distributed in several stages. The first is to discover how much another camera is an ally or opponent. Cameras with more similar view points are opponents and will attempt to bias each other from looking at the same object. However, cameras with dissimilar view points are allies and attempt to bias other allies towards what they are looking at. This can be accomplished for instance by analyzing feature vectors from each camera. The more dissimilar feature vectors from each camera are over time, the more dissimilar their viewpoints are.

At this point, it should be noted that the definition of an ally or opponent is not Boolean nor mutually exclusive. An example of this can be seen in figures 4,5 and 6. Using neural principles, two agents work in a manner to suppress or excite features between one another. For instance, a camera will attempt to suppress the saliency of feature vectors greatly for cameras with similar view points, but may also excite those feature vectors, but very little. The net effect would be suppressive. This means that some camera agents may have view points that are sort of similar, but the net effect of excitation and suppression will be zero. Figure 5 shows the effect of this divided attention.

In order for this system to work, not only must agent cameras be able to bias each other, but they must maintain a task based working memory. This is because at any one instant, two or more cameras may not be looking at the same thing. If bias is stateless, cameras will be less effective at biasing each other since what one camera was looking at a second ago is now forgotten and cannot bias another camera. To create a state we draw upon models of working memory [5,6,7]. The idea is that camera agents will maintain a residual memory. Thus, what a camera observes in any instance will bias other camera agents over an interval. Since information in general becomes stale, so should the memory. This is accomplished by allowing the memory to leak. Thus, a camera agent's bias for a feature becomes like a capacitor. As a feature is observed, potential is gained. If a feature is not observed for a significant interval, potential is lost. This allows a camera agent to continue biasing other camera agents to or from certain features even when it is not currently observing those features. Attention can now be biased with a rudimentary state.

### 3.2.4 Rudimentary Object Identification

The visual saliency program collects 42 feature vectors about an object at multiple scales. Elementary features such as colors and angles can be used to convey information about the identity of objects. While not a comprehensive method of identifying objects, it can begin to give the network an idea of how certain objects tend to behave. That is, it can begin to look for patterns. For instance, humans have a rather distinct color of skin. This rudimentary feature can give the network ideas about what is happening in the room. For instance, some human is standing where I believe the photo copier is or some human is at the door location. Certain contextual assumptions and probabilistic inferences can allow the network of agents to know the likelihood of two events co-occurring. If an event happens that is rare, it may be an event that is worth attending to based upon its novelty. Additionally, certain heuristics can be programmed in, for instance no one is to enter the room between 10pm and 7am. If a human is at the door between these hours, it should be deemed an interesting event.

The sort of object identification described has been developed in our lab and tested using two different schemes. The first takes a set of feature vectors and clusters them together using a combination of water level clustering with multi-scale voting. This works by placing features in space then isolating individual clusters by lowing the "water" and creating "islands" of clustered feature vectors. Since features are extracted at multiple scales, each scale is

initially scaled separately. The decision as to whether a cluster of feature vector constitutes an object is identified by voting between the scales. This algorithm attained an 80% accuracy at identifying between four different objects (coke can, fire plug, handicap sign, campus info sign) in natural scenes.

Another clustering method was also developed. This one involves training the algorithm to cluster according to the subjective judgment of the experimenter. Thus, the chore was not only to cluster feature vectors into objects, but was also to learn how to do this. The algorithm works by linking feature points whereby a point is linked to another that is close, but is in a denser region. To create segmented objects, the links are cut. The choice as to which links are cut is based upon the statistics of the links in the feature space along with what properties that are used to distinguish which links to cut. The properties for cutting are determined by training that uses sequential quadratic programming to hill climb until the program is creating segmented objects in a manner the agrees as closely as possible to what the researcher believes is a good segmentation. Accuracy for this method reached 89% with the same image set as the first algorithm, but is in general slower.

These two algorithms[8] both demonstrate that feature vectors can be used to create rudimentary distinctions between objects. Additionally, one of the methods works in real time, the other works very quickly and may be optimized into real time. Additionally it should be noted what is different between the approaches mentioned and current clustering or feature vector identification schemes. The first is that the number of clusters does not need to be pre-specified nor does the size and shape need to be either as is the case with classical methods such as expectation maximixation, K-Means etc. Thus, both methods work where the shape and number of clusters is uncertain or even initially unknown. The methods also differ from other flexable methods such as Kohonen maps in that very few iterations are needed and classification is done quickly. Thus, feature vector clustering for redamentry object identification in our paradigms is both fast and flexable.

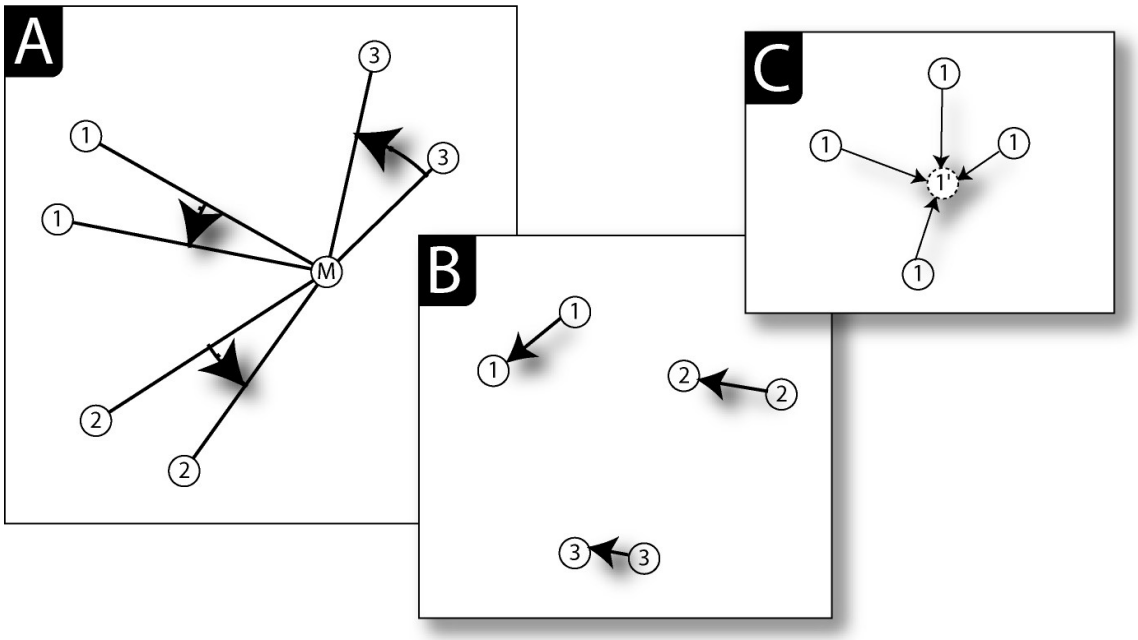


Figure 7: A universal map is created from individual maps. Since there is no universal coordinate reference, steps A and B are necessary. A) Two or more maps are matched first by aligning them at their center of mass (M). The first step is to rotate the maps such that camera distances are reduced as much as possible. That is map 1 is rotated against map 2 until for instance camera 1 in map 1 is as close as possible to camera 1 in map 2. The amount of rotation is the mean error in rotational offset for all cameras from the counterpart in the other map. B) The same procedure is repeated for translation of the map coordinate frames. C) The final map is the differential of all the maps. That is 1' is the average of the locations of the location of camera 1 on all the maps.

[8] Some information can be found at http://www.nerd-cam.com/misc/cluster.pdf

### 3.3 Learning to self-calibrate

One goal of our project is to create a self-organizing network that does not require large amounts of intervention. That is, we would like for the system to be easy enough to use such that if deployed in the market, most people will be able to configure and run it. In order to accomplish this, the network must be able to be configured and run by most people, not just engineers. Thus, self-calibration of the camera network is advantageous since manual calibration requires technical expertise. We are exploring several methods for doing this. One such method takes advantage of the fact that the camera on the NC-1 when moved, moves the projection plane. This means that in order to get a stereoscopic reading on a distance, the NC-1 can take one picture, rotate and take another. This establishes depth by triangulation with a single camera. To accomplish this, the potent motion channel in the visual saliency program is used. In a simple scenario, to calibrate camera A to camera B, you would first have camera B move vigorously and constantly. Camera A would find B quickly and would then proceed to take several parallax images of B. When done, the roles would reverse and A would move vigorously while B measures where A is. By involving several cameras in this network we gain strength in several ways. First, the maps each camera creates can be compared. A more accurate universal map can then be made from the differential. Figure 7 describes this process.

Additionally, repeated measures can establish a camera's IQ. This is, if a camera has consistently poor calibration, several calibration runs should show it to have higher error when comparing differentials. That is, when the maps are brought together to make a universal map, each individual map will have an error when compared with the universal map. Cameras with a statistically high error for this comparison are less trustworthy. By rooting them out, when the universal map is created, cameras with a low IQ will be weighted lower than cameras that have proven themselves to be more accurate in creating the differential.

### 3.4 Integration of large scale systems with wrappings

An important part of our system is the integration of its multiple components. This is a difficult task in general for

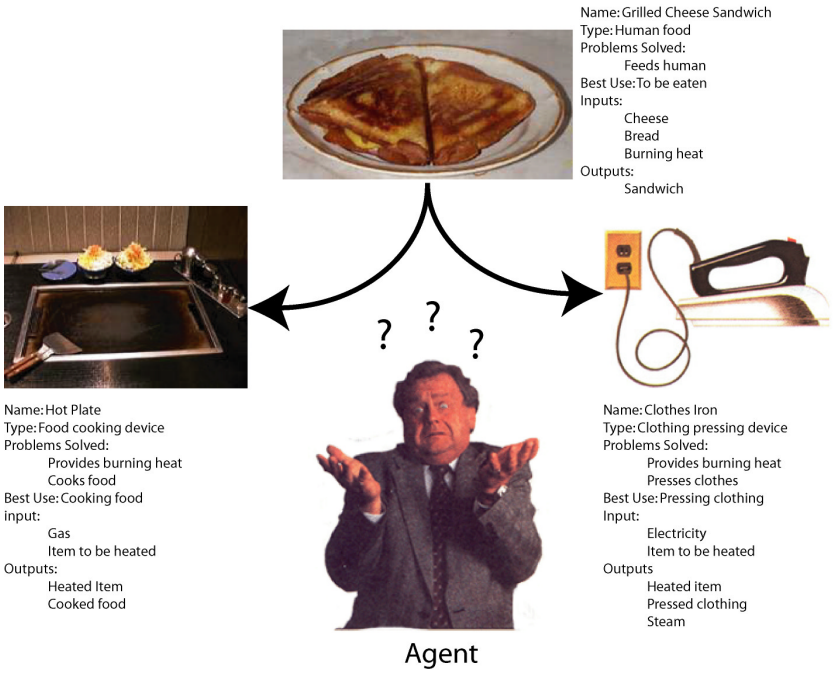


Figure 8: The intelligent agent inspects the wrappings for each resource and makes a decision as to resource usage. For instance in this example, the hot plate is a superior resource for making a grilled cheese sandwich. However, in its absence, a clothes iron could fulfill this purpose. This extends into many complex systems where the best resources to complete a job must be matched up.

several reasons. The first is that it is difficult to ensure that all the systems will “play nice” together. That is, as information and tasks are passed from one entity to another, how do we make sure that the tasks and information are compatible across the whole system? Another difficulty is that of adding entities to a large-scale system. How do you ensure that the new system is backwards compatible with other entities it may need to interface with? Additionally, if one entity is removed, but another entity is available that might be able to complete its task; you want some sort of switch over. Thus, in a large-scale system you have tricky logistical problems with interfacing entities and substituting for entities to maintain the system “up”.

One successful approach to this problem has been what is known as wrappings [10,11]. The idea is that each entity in the system is treated as a resource to be used and connected as needed. Each resource has things it can do, things it needs, things it was designed to do. The basic idea is to define all resources in a system, to the best of your ability, such that an agent, which is also a resource, can know how to connect and juggle resources. For instance, in figure 8, both an iron and a hot plate can make a grilled cheese sandwich. With wrappings you would specify the iron and hot plate as resources that can do things. For instance, both an iron and the hot plate will heat things to a burning temperature, but the hot plate is made to heat food. The iron is made to heat clothes. The grilled cheese sandwich is also a resource. It is food, and it needs to be heated to burning temperatures. An agent would look and find that a hot plate and an iron are present. Looking at what is provided, the agent chooses the hot plate to make the grilled cheese sandwich because while both the iron and the hot plate can heat things to a burning temperature, the hot plate is specified to be for food. In the absence of the hot plate, the agent will choose the iron. This is because the iron can provide what is needed in this case, namely burning heat. Thus, with wrappings we do the same for hardware and software systems. We take individual resources and designate what they need and what they will provide. We also designate what they’re made for. An agent’s job is to match these things up such that needs are provided for by resources that can provide them. Additionally, humans are considered as a resource. When in doubt, the agent can ask a human to provide something for a resource, for instance it can ask a person to plug something in. A human can also be used as a resource if the agent needs knowledge to solve a problem. Thus, the human as a resource provides information.

As resources are wrapped in this manner, they can be connected by the agent automatically. This allows very large scale systems to be connected automatically, and failed resources to be substituted on the fly. For our surveillance network this means for instance, that we can supply a variety of camera types and CPU types. The wrapping agent will know the abilities of different cameras and computers. If a camera is removed that performs a certain task, the agent can select another camera that also can perform that task, in the processes selecting a camera, which is best suited for that role.

## 4. DISCUSSION

This is a work in progress. Many of the components are complete while many are still being actively completed. It is important to note which components are reality and which are not. The iLab Neuromorphic Tool kit, NSL and wrappings are well-developed technologies. The Beowulf saliency program is also in a mature state. Eight NC-1 cameras have been completed and integrated into our 16 CPU Beowulf cluster. Intrinsic calibration as well as control mechanisms are complete for the NC-1. Objects identification by several clustering techniques is also complete. Simulations of a distributed attention in NSL have been written but are awaiting complete testing. Components that still need completion, but are in active development are distributed NSL, camera network calibration and attention balancing in the network. It is expected that by publication date the camera network calibration and distributed NSL will be completed.

## 5. CONCLUSION

Results from completed components have demonstrated that our system attends to salient objects in a natural scene in real time using two degree of freedom cameras. It can also infer some object recognition from the feature vectors. It may also be trained how to better recognize objects based upon the researchers subjective categorization of objects. Current research is concentrating on camera network mapping and calibration, divided attention and executive control.

## ACKNOWLEDGEMENTS

I would like to thank Mike Olson for his help and suggestions. This research is supported by the National Imagery and Mapping Agency, the National Science Foundation, the National Eye Institute, the Zumberge Faculty Innovation Research Fund, the Charles Lee Powell Foundation and Aerospace Corporation.

## REFERENCES

1. Weitzenfeld A, Arbib M A, Alexander A, The Neural simulation Language: A system for brain modeling, 2002, The MIT Press, Cambridge, Massachusetts
2. Itti L, Koch C, 2000, A saliency-based search mechanism for overt and covert shifts of visual attention, *Vision Research*, **40(**10-12): 1489-1506.
3. Itti L, Koch C, 2001, Computational Modeling of Visual Attention, *Nature Reviews Neuroscience*, **2(**3): 194-203.
4. Itti L, Gold C, Koch C, 2001, Visual Attention and Target Detection in Cluttered Natural Scenes, *Optical Engineering*, **40(**9): 1784-1793.
5. Durstewitz D, Seamans J K, Sejnowski, 2000, Neurocomputational models of working memory, *nature neuroscience*, **3**(2000):1184-1191
6. Tanaka S, 2000, Computational approaches to the architecture and operations of the prefrontal cortical circuit for working memory, *Prog. Neuro-Psychopharmacol. & Biol. Psychiat*, **25**:259-281
7. Rougier N P, O'Reilly R C, 2002, Learning representations in a gated prefrontal cortex model of dynamic task switching, *Cognitive Science*, **26**: 503-520
8. Lyons M J, Harrison L G, 1992, Stripe Selection: An intrinsic property of some pattern-forming models with nonlinear dynamics, *Developmental Dynamics*, **195**: 201-215
9. Oztop, E., and Arbib, M.A., 2002, Schema Design and Implementation of the Grasp-Related Mirror Neuron System, *Biological Cybernetics*, 87:116-140.
10. Landauer C, Bellman K L, "*Model-Based Simulation Design with Wrappings*", pp. 169-174 in Proceedings of 00S'97: Object Oriented Simulation Conference, WMC'97:1997 SCS Western Multi-Conference, 12-15 January, Phoenix, SCS International (1997)
11. Landauer C, Bellman K L, "*Wrappings for Softwm'e Development*", pp. 420-429 in 1st Hawaii Conference on System Sciences, Volume III: Emerging Technolo- gies, 6-9 January 1998.