Toward highly capable neuromorphic autonomous robots: beobots

Laurent Itti

Department of Computer Science, University of Southern California, Los Angeles, CA

ABSTRACT

We describe a new mobile robotics platform specifically designed for the implementation and testing of neuromorphic vision algorithms in unconstrained outdoors environments. The new platform includes significant computational power (four 1.1GHz CPUs with gigabit interconnect), a high-speed four-wheel-drive chassis, standard Linux operating system, and a comprehensive toolkit of C++ vision classes. The robot is designed with two major goals in mind: real-time operation of sophisticated neuromorphic vision algorithms, and off-the-shelf components to ensure rapid technological evolvability. A preliminary embedded neuromorphic vision architecture that includes attentional, gist/layout, object recognition, and high-level decision subsystems is finally described.

1. INTRODUCTION

Animals demonstrate unparalleled abilities to interact with their natural visual environment, a task which remains embarrassingly problematic to machines. Obviously, vision is computationally expensive, with a million distinct nerve fibers composing each optic nerve, and approximately half of the mammalian brain dedicated more or less closely to vision.[6] Thus, for long, the poor real-time performance of machine vision systems could be attributed to limitations in computer processing power. With the recent availability of low-cost supercomputers, such as so-called "Beowulf" clusters of standard interconnected personal computers, however, this excuse is rapidly losing credibility. What could then be the reason for the dramatic discrepancy between animal and machine vision? Too often computer vision algorithms are designed with a specific goal and setting in mind, e.g., detecting traffic signs by matching geometric and colorimetric models of specific signs to image features.[3] Consequently, dedicated tuning or algorithmic alterations are typically required to accommodate for novel environments, targets or tasks. For example, an algorithm to detect traffic signs from images acquired by a vehicle-mounted camera will typically not be trivially applicable to the detection of military vehicles in overhead imagery.

Much progress has been made in the field of visual neuroscience, using techniques such as single neuron electrophysiology, psychophysics and functional neuroimaging. Together, these experimental advances have set the basis for a deeper understanding of biological vision. Computational modeling has also seen recent advances, and fairly accurate software models of specific parts or properties of the primate visual system are now available, which show great promise of unparalleled robustness, versatility and adaptability. A common shortcoming of computational neuroscience models, however, is that they are not readily applicable to real images.[6] *Neuromorphic engineering* proposes to address this problem by establishing a bridge between computational neuroscience and machine vision. An example of neuromorphic algorithm developed in our laboratory is our model of bottom-up, saliency-based visual attention,[8,6] which has demonstrated strong ability at quickly locating not only traffic signs in 512x384 video frames from a vehicle-mounted camera,[7] but also — without any modification or parameter tuning — artificial targets in psychophysical visual search arrays,[5] soda cans, emergency triangles,[7] faces and people,[10] military vehicles in 6144x4096 overhead imagery,[4] and many other types of targets **(Fig. 1)**.

The new robotics platform described here is a test-bed aimed at demonstrating how neuromorphic algorithms may yield a fresh perspective upon traditionally hard engineering problems, including computer vision, navigation, sensorimotor coordination, and decision making under time pressure. This contrasts with the motivation behind biorobots,[15,12] which aim at physically and mechanically resembling animal systems. To exploit real-time video streams and effectively base control on computationally-demanding neuromorphic vision algorithms, our new robots combine a small Beowulf cluster to a low-cost but agile four-wheel-drive robotics platform, together forming a "Beowulf-robot" or Beobot.

What will Beobots be capable of that existing robots cannot already achieve? Most robots have under-emphasized the vision component that is our main focus, and rely instead on dedicated sensors including laser range finders and sonars. Much progress has been made in developing very impressive *physically capable* robots (e.g., Honda humanoid).

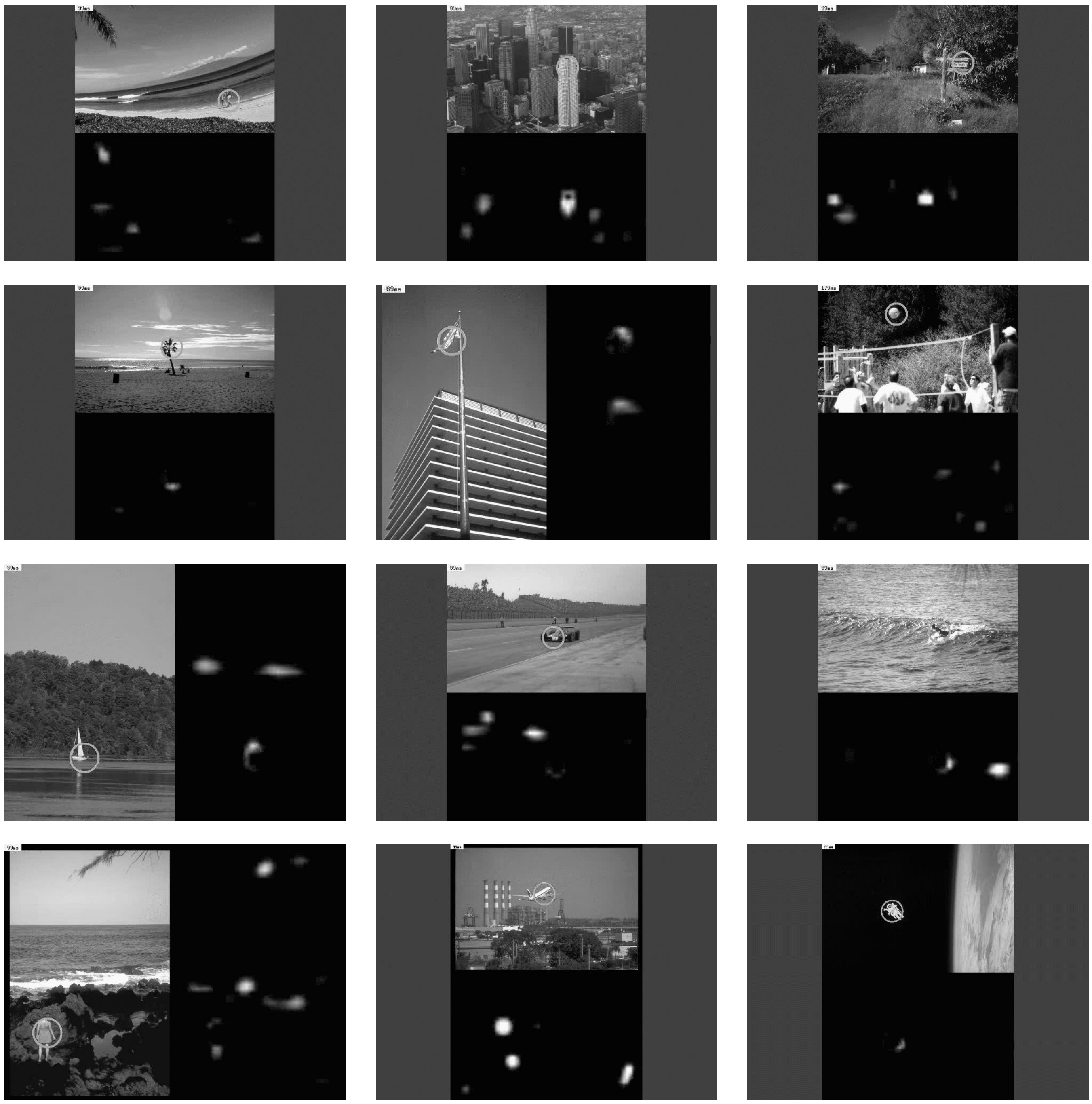

Figure 1. Example images and the associated most salient object detected by our model of bottom-up, saliency-based visual attention (marked by a circle). Images are shown with the associated saliency map (brighter locations in the saliency map are more salient). Note how the attended location (circle) has been erased from the saliency map, through an inhibition-of-return mechanism which will allow the system to attend next to the second most salient location. From left to right and top to bottom, the algorithm attended first to: a couple walking on a beach; a building; a sign; a palm tree; a flag; a volley ball; a boat; a race car; a surfer; a woman; a plane; and a space-walker. For a clearer rendering of these samples, please see `http://iLab.usc.deu/bu/movies/`, the ninth video. These and many other sample images demonstrate the generality of the algorithm, which we believe makes it uniquely suited to operation in unconstrained environments.

In addition, very sophisticated and powerful algorithms are now available that make robots *intelligent*.[16] However, we believe that some improvement still is possible in making robots more *visually capable*, as current systems often rely on simplified, low-computation visual tricks which greatly limit their autonomy.

Below we describe the basic hardware and software components of the Beobots, as implemented in a working prototype. We further describe a preliminary software system that builds upon these components, and implements a neuromorphic vision architecture that includes visual attention (modeled after processing in the dorsal visual stream of the primate brain), localized object recognition (modeled after processing in the primate ventral stream), rapid computation of the gist/layout of the scene, and high-level decision for basic navigation.

At the stage of development described, which is the successful creation of a working Beobot prototype, the present paper is limited to a fairly technical description of the various elementary hardware and software system. We hope, however, that the overall approach described here may trigger some useful discussion with respect to the feasibility of embedding neuromorphic vision algorithms onto a robotics platform. For further insight on the type of new algorithms that may become realizable with the availability of the Beobot platform, we refer the reader to, for example, ref.[11]

2. THE ROBOTICS PLATFORM

In this section we briefly describe the hardware components of the Beobots, as implemented in the prototype shown in **Fig. 2**. The development of a new robotics platform is justified by the current unavailability of any reasonably-priced commercial platform with processing power suitable for real-time neuromorphic vision. Guidelines for our design included:

- High-speed, agile chassis, at the cost of precision and ease of control;
- Standard off-the-shelf components that can easily be replaced or upgraded;
- Compatibility with open-source software and familiar development tools;
- Low cost of individual parts and of the complete assembly;
- Small size for ease of use and maneuverability.

It is important to note a few key differences between Beobots and existing, similarly-looking robots. A primary goal for Beobots, which may or may not turn out to be achievable, is *autonomous operation in unconstrained environments.* This directly contrasts with remotely-operated robots where computation is performed on a central server communicating with the robot via a radio link (e.g., Clodbuster robots[2]), with semi-autonomous robots which require overall guidance from a human operator but can shape this guidance according to environmental conditions (e.g.,[9]), and with robots operating in constrained environments such as an artificial soccer field. The extent to which fully autonomous operation will be achievable will most probably depend on task difficulty (e.g., going to the library to pick up books is more difficult than the first test task described below, running around a track). However, it is our goal for Beobots to avoid developing algorithms that are task-specific, and rather develop a number of biologically-inspired computational modules. As mentioned in introduction, the existing bottom-up attention module is an example of such component that has been sccessfully applied to a very wide, unconstrained range of inputs.

2.1. Embedded Beowulf Cluster

The Beowulf in our Beobot prototype is a standard double dual-CPU embedded Linux system with Gigabit Ethernet link between both dual-CPU boards. We use 1.1 GHz Pentium-III (PIII) CPUs, which may be upgraded to faster models as available. The motherboards rest on a custom-built 3-layer mounting platform, composed of a 5mm-thick laser-cut base plate made of bulletproof Lexan material, a 5mm-thick laser-cut rubber layer for firm yet shock-insulated resting of the motherboards, and a 1mm-thick laser-cut acrylic protective cover. Any motherboard with the standard PICMG form factor can be installed onto the CPU platform. We used two ROCKY-3742EVFG motherboards, as these integrate all of the peripherals required for our application, including: Support for dual Pentium-III CPUs, connector for solid-state (256MB CompactFlash) hard-disks, on-board sound for voice recognition and synthesis, on-board FireWire port for video capture, on-board Gigabit Ethernet for interconnection between both boards, and on-board 10/100 Ethernet for connection to host computers during software development.

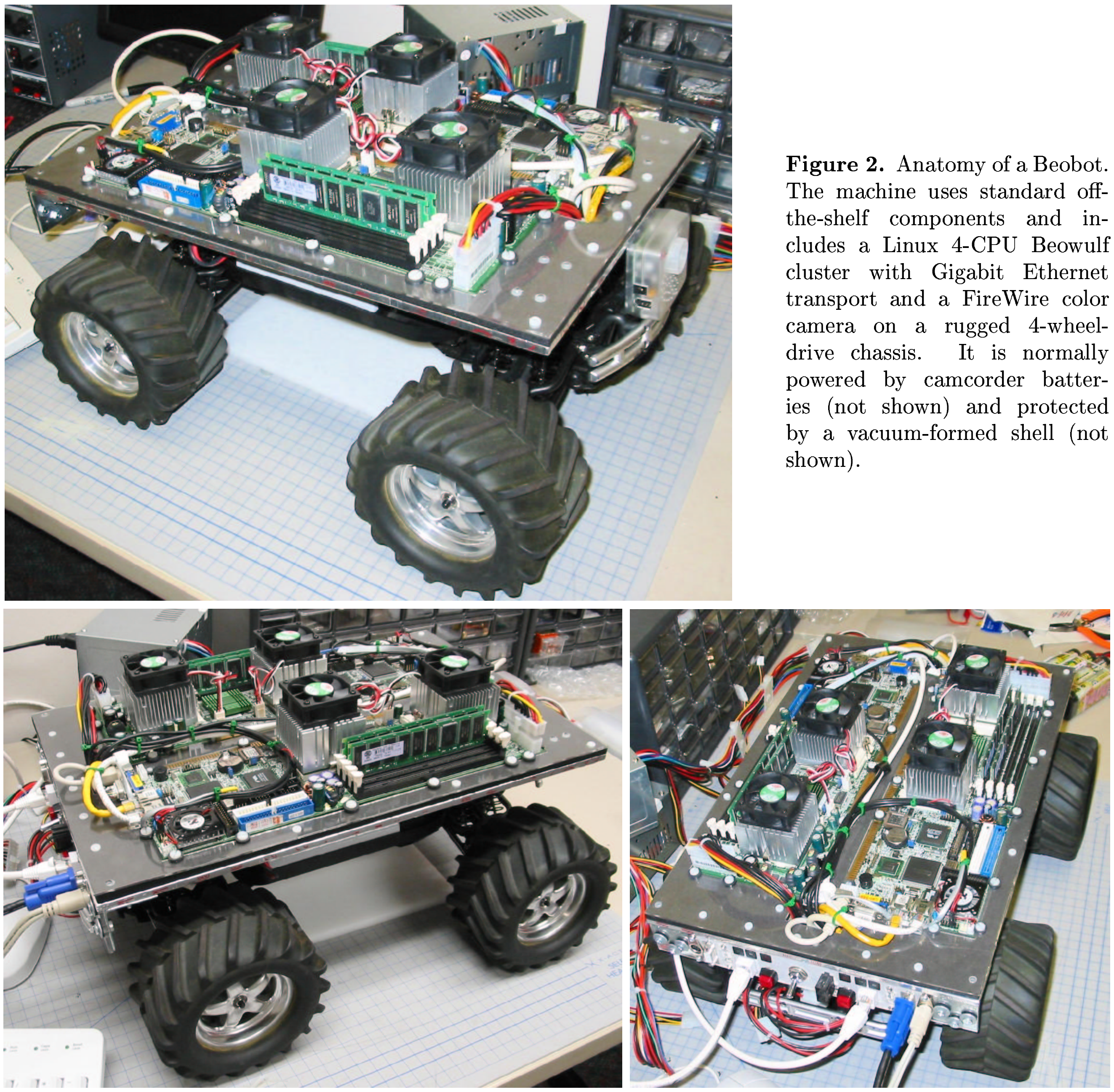

Figure 2. Anatomy of a Beobot. The machine uses standard off-the-shelf components and includes a Linux 4-CPU Beowulf cluster with Gigabit Ethernet transport and a FireWire color camera on a rugged 4-wheel-drive chassis. It is normally powered by camcorder batteries (not shown) and protected by a vacuum-formed shell (not shown).

The Beowulf cluster drains a maximum of 30A at 5V and 2A at 12V, which are provided by eight standard R/C battery packs (for an autonomy of approximately one hour) or eight high-capacity Lithium-Ion battery packs (for an autonomy of approximately two hours).

2.2. Mobile Platform

We have chosen to use a radio-controlled (R/C) vehicle as basis for the Beobots **(Fig. 2)**, although these have overall poor reputation in the robotics community: indeed, they are optimized for speed and light weight, at the expense of accuracy in control. Yet in many respects they resemble full-size vehicles, which humans are able to control at high speed, without requiring laser range finders, wheel encoders and other typical robotics artifacts.

The Traxxas E-Maxx platform (www.traxxas.com) was well suited as the basis for Beobots. With dual high-torque electric motors, it can handle the additional weight of the CPUs and batteries. With a top speed of 25 MPH and an autonomy of 20 minutes on standard R/C NiMH battery packs, it is ideal for fast, ballistic operation and control, similar to the control we exert while driving real automobiles. The radio control is equipped with a high/low gear shift switch, which we have used to instead switch between autonomous and human radio-controlled modes (ideal for online learning). Thus, while the robot usually operates autonomously, it is possible for a human operator to easily override this behavior (e.g., in case of an imminent accident). A serial to pulse-width-modulation module is used to control the servos (steering, speed control with brakes, and 2-speed gearbox) from the on-board computers (see, e.g., www.seetron.com/ssc.htm). A speed sensor linked to the drive train is being developed to obtain speed estimates based on the mechanisms found in standard computer mice. The shocks have been stiffened and the tires filled with firm foam such as not to collapse under the payload.

For ease of connection to a host computer and to various equipment, the keyboard, mouse, video, USB, and Ethernet ports have all been deported to a single connection panel at the back of the Beobot. This panel also includes an external 15V/12A power connector, and a switch to select between external and battery power. Software has been developed to connect a small LCD screen to one of the serial ports of the robot, which will be mounted on the final protective shell over the motherboards. The cluster can accept a variety of accessory equipment, including GPS (connected to a serial port), wireless networking (through a USB port), additional hard-disk drives (through the IDE ports), and virtually any other standard PC peripheral that is supported by the Linux operating system.

3. THE C++ NEUROMORPHIC VISION TOOLKIT

The cluster runs a standard Linux distribution, which is simply installed from CD-ROM onto the 256MB CompactFlash solid-state disk of each motherboard. For our prototype, we used the Mandrake 8.2 distribution, which automatically detected and configured all on-board peripherals, except that an alternate Gigabit Ethernet driver was necessary for proper operation. To allow rapid development and testing of new neuromorphic vision algorithms, we are developing a comprehensive toolkit of C++ classes.

The toolkit provides a number of core facilities, which include:

- Reference-counted, copy-on-write memory allocation for large objects such as images (so that various copies of an image share the same physical memory until one of the copies attempts to modify that memory, at which point a copy of the memory is first made);
- Template-based classes, so that objects such as images or image pyramids can be instantiated with arbitrary pixel types (e.g., scalar byte or float pixels, color double pixels, integrate-and-fire neuron pixels, etc);
- Automatic type promotion, so that operations among template classes automatically avoid all overflows (e.g., multiplying an image of bytes by a float coefficient results in an image of floats);
- Automatic range checking and clamping during demotion of types (e.g., assigning an image of floats to an image of bytes transparently converts and clamps all pixel values to the 0..255 range);
- Smart reference-counted pointers, so that when the last pointer to an object is destroyed, memory allocated for the pointee is automatically freed;
- A convenient logging facility to report debugging and other messages to standard output, LCD screen or system logs.

Building on these core elements and concepts, the basic facilities provided by a set of generic C++ classes in the toolkit include:

- Low-level graphic elements such as 2D point, RGB pixel, rectangle, etc;
- A template Image class that defines a copy-on-write 2D array (of data type chosen through the C++ template mechanism) and provides numerous low-level image processing functions, such as convolution, rescaling, arithmetic operations, decimation & interpolation, various normalizations, drawing primitives, miscellaneous functions such as speckle noise, 3D warping, flooding, edge detection, 3/4 chamfer distance transforms, and finally neuromorphic operations including center-surround and retinal filtering;
- A template Image Pyramid class which implements dyadic pyramids of various data types (defined as template argument) and for various pyramid schemes,[1] including Gaussian, Laplacian, Gabor, and Template Matching;

- Classes for storage, retrieval and display of Images in various file formats;
- Several classes specific to our model of bottom-up, saliency-based visual attention, including a Visual Cortex class (contains a run-time-selectable collection of pyramids, including for color, intensity, orientation and flicker information), a Saliency Map class (2D array of leaky integrate & fire neurons), a Winner-Take-All class (distributed neuronal maximum detector), an InferoTemporal class (with run-time selectable object recognition scheme, including backpropagation and HMAX[14]), a Brain class (contains a retina, visual cortex, saliency map, winner-take-all and a few other objects);
- Several classes specific to our model of contour integration in the primate brain, which simulates intricate patterns of connections between neurons visually responsive to various visual locations;
- A Jet class (vector of responses from neurons with various tuning properties but at a same location in the visual field), used by our ImageSpring model that rapidly segments a rough layout from an incoming scene;
- Classes to capture video frames through PCI, USB and FireWire interfaces, and to capture audio (including decoded radio-control signals);
- Classes to read/write configuration files and to manage internal program options (from configuration files or command-line arguments);
- A set of fast multi-threaded interprocess communication classes which allow quick transfer of images and other data from one CPU to another, using either TCP/IP or shared memory. These include a Beowulf class that automatically sets up interconnections between different computers on a network, and transparently handles sending and receiving of messages;
- Several accessory and Beobot-specific classes, such as Timer, XML parser, interface to LCD screen, and interface to servomechanisms.

Building on these core facilities, a number of additional classes and executable programs have been developed, to process movie sequences over a Beowulf cluster, to control the Beobots, and to implement various models of attention, contour integration, object recognition, scene layout computation, high-level scene interpretation, etc.

4. RESULTS

A preliminary application is being developed for testing of the Beobots with a simple task: drive as fast as possible along the USC Olympic running track, avoiding obstacles such as joggers. The architecture used for this purpose shares some similarity to Rensink's triadic architecture of human vision,[13] relying on: a rapid computation of scene layout, to localize the track; low-level visual processing that guides visual attention bottom-up, to locate obstacles; localized object recognition to identify obstacles and other salient scene elements being attended to; high-level decision based on a working memory of recent percepts and actions; and interfacing with the robot's electromechanical actuators **(Fig. 3)**.

The application is being developed and refined with encouraging results **(Fig. 4)**. While layout and saliency are robustly computed in most situations, object recognition often is more problematic, especially when background clutter is present. Nevertheless, this simple application is a working example of how distributed neuromorphic architectures may be developed on Beobots using our C++ vision toolkit, for real-time outdoors operation.

5. DISCUSSION AND OUTLOOK

With the successful development of a prototype Beobot for a total cost below $5,000, we have shown how standard off-the-shelf PC and R/C components could be assembled to yield a robotics platform suitable to the real-time operation of neuromorphic vision algorithms. While evolvability typically is a major issue in robotics design, Beobots can be upgraded in minutes to faster CPUs, faster or better PICMG motherboards, new USB, FireWire, serial, IDE or other peripherals, any faster or more powerful R/C chassis that uses standard R/C servomechanisms, new Linux distributions, and new application software.

Blueprints for the custom-designed components of the Beobots (CPU mounting platform, protective shell, and battery power conversion module) are being made available through our web site at `http://iLab.usc.edu/beobots/`.

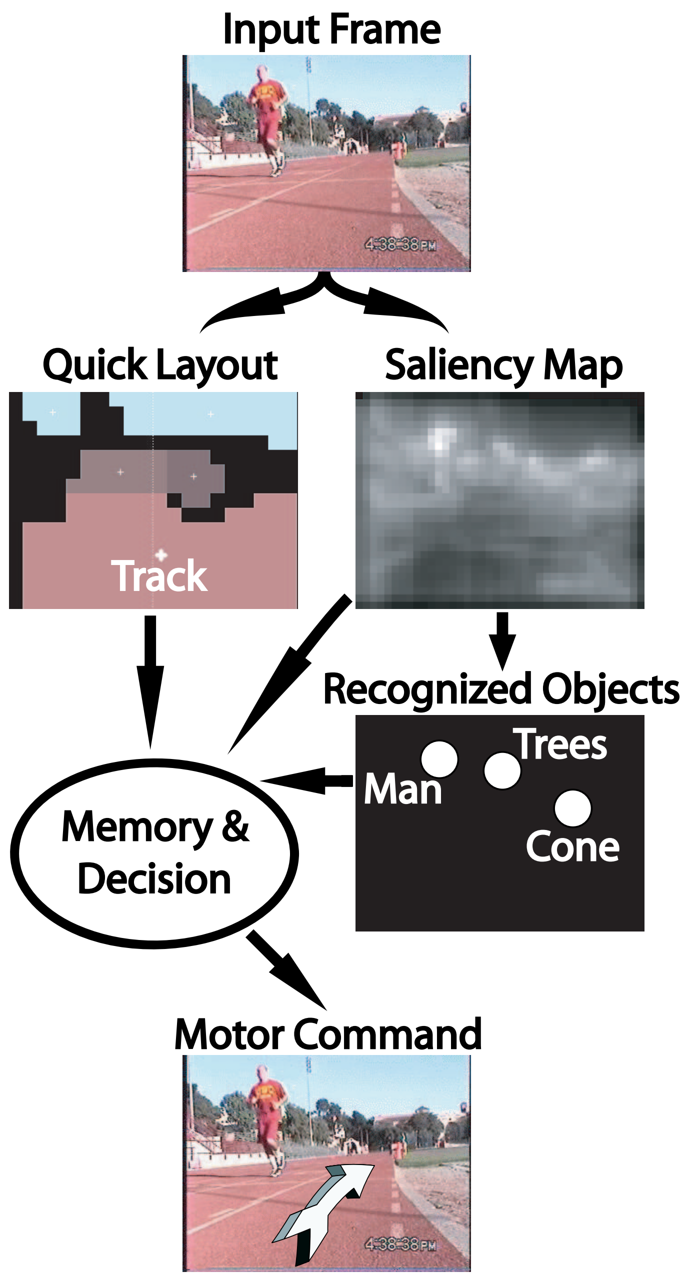

Figure 3. A prototype distributed neuromorphic vision application developed for the Beobots. A very rough layout of each incoming frame is computed, and the road is located as the largest region in the lower half of the image. In parallel, low-level computation of early visual features (color opponencies, intensity contrast, oriented edges, and motion transients) is distributed over the four CPUs of the Beobot. A non-linear combination of the resulting feature maps yields the topographic saliency map (brighter regions indicate more salient locations). The saliency map guides focal visual attention, which selects objects from the image in order of decreasing saliency. Each selected object is passed to an object recognition module which attempts identification, with variable success depending on background clutter. Based on the current and past layouts, saliency maps, sets of recognized objects, and on the goal assigned to the robot, a rule-based agent determines the next action. This action is finally communicated to the motor components of the robots, after some smoothing and possible radio-control override.

The source code for the C++ toolkit is already available through CVS access and at `http://iLab.usc.edu/toolkit/`. A discussion forum around this project and other neuromorphic models is also available through this web page.

A number of enhancements are being studied, including alternate models of localized objet recognition, voice recognition and synthesis, and algorithms for high-level scene understanding, navigation and planning. Although all software components currently are synchronized by the video rate of 30 frames/s, a smarter scheduler is also being studied to balance the computational load across the four CPUs, and allow subsystems running at different time scales to continuously exploit all computing resources. This would, for instance, allow the robot to perform object recognition asynchronously from the computation of salience and other low-level visual processing.

In summary, our approach directly follows the recent revolution brought to the high-performance computing community by Beowulf clusters, replacing costly and slowly-evolving custom CPUs and bus architectures by low-cost assemblies of mass-produced, rapidly-updated PC components. Based on our first prototype, we believe that the Beobot approach has potential for making the implementation of sophisticated neuromorphic algorithms onto robots

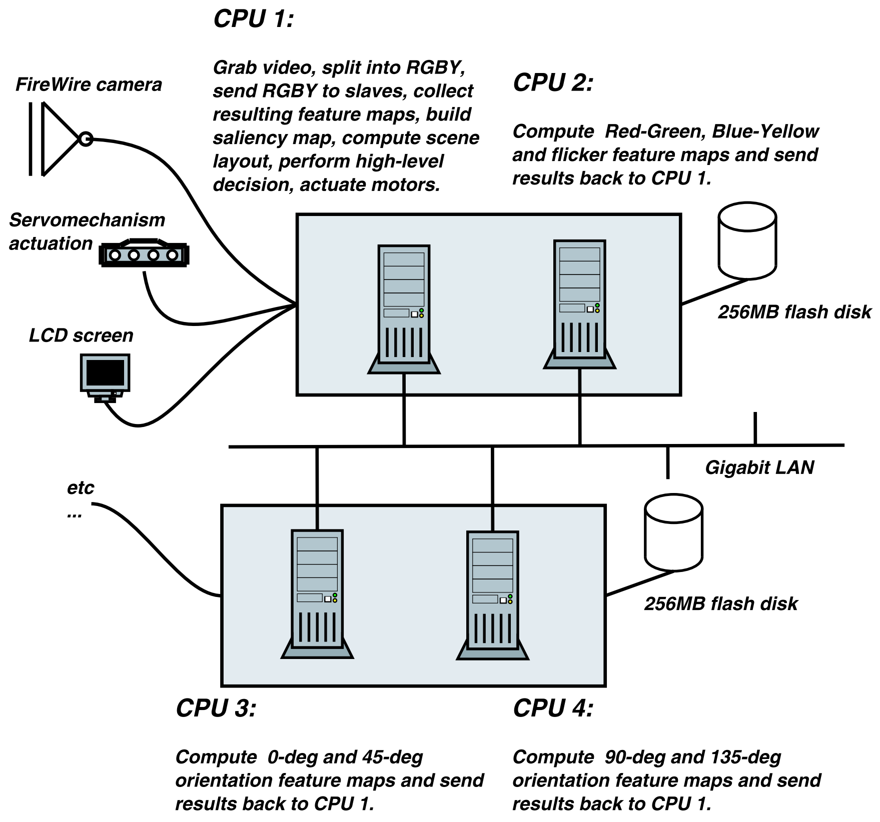
CPU 1:
Grab video, split into RGBY, send RGBY to slaves, collect resulting feature maps, build saliency map, compute scene layout, perform high-level decision, actuate motors.
CPU 2:
Compute Red-Green, Blue-Yellow and flicker feature maps and send results back to CPU 1.
FireWire camera
Servomechanism actuation
LCD screen
256MB flash disk
Gigabit LAN
etc ...
256MB flash disk
CPU 3:
Compute 0-deg and 45-deg orientation feature maps and send results back to CPU 1.
CPU 4:
Compute 90-deg and 135-deg orientation feature maps and send results back to CPU 1.

Figure 4. Implementation of the prototype Beobot application on the 4-CPU embedded cluster. Because dual-CPU boards are used, mapping of vision subsystems onto CPUs actually is not as clearly defined as shown here: on each dual-CPU board (grey boxes), two processes run simultaneously, and the Linux operating system allocates them to the two available CPUs of each board. Shared memory is used to pass messages between processes running on a same board, and TCP/IP transport over the embedded Gigabit Ethernet network is used to pass messages across the two boards. Various peripherals are connected to the available serial, USB, FireWire, parallel, audio, keyboard, mouse, ethernet and IDE ports on both motherboards, including video cameras, LCD creeen, R/C car servomechanism actuators, etc. For additional details, please see `http://iLab.usc.edu/beobots/`.

a reality. The challenge which lies ahead will now be to adapt more general neuromorphic vision algorithms (such as, e.g.,[11]) for real-time operation on the Beobots.

Acknowledgments: This work is supported the National Science Foundation, the National Eye Institute, the National Imagery and Mapping Agency, the Zumberge Research and Innovation Fund and the Charles Lee Powell Foundation.

REFERENCES

1. P J Burt and E H Adelson. *IEEE Trans on Communications*, 31:532–540, 1983.
2. A Das, R Fierro, V Kumar, J Southall, J Spletzer and C Taylor, In *Proc IEEE Int. Conf. on Robotics and Automation,* Seoul, Korea, pp. 1714-1719, 2001.
3. A de la Escalera, L E Moreno, M A Salichs, and J M Armingol. *IEEE Trans Ind Elec*, 44(6):848–859, 1997.
4. L. Itti, C. Gold, and C. Koch. *Optical Engineering*, 40(9):1784–1793, Sep 2001.
5. L. Itti and C. Koch. *Vision Research*, 40(10-12):1489–1506, May 2000.
6. L. Itti and C. Koch. *Nature Reviews Neuroscience*, 2(3):194–203, Mar 2001.
7. L. Itti and C. Koch. *Journal of Electronic Imaging*, 10(1):161–169, Jan 2001.
8. L. Itti, C. Koch, and E. Niebur. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, Nov 1998.
9. L Matthies, Y Xiong, R Hogg, D Zhu, A Rankin, B Kennedy, M Hebert, R Maclachlan, C Won, T Frost, G Sukhatme, M McHenry and S Goldberg. In *Proc. of the 6th International Conference on Intelligent Autonomous Systems,* Venice, Italy, Jul 2000.
10. F. Miau and L. Itti. In *Proc. IEEE Engineering in Medicine and Biology Society (EMBS), Istanbul, Turkey*, Oct 2001.

11. V. Navalpakkam and L. Itti. In: *Proc. 2nd Workshop on Biologically Motivated Computer Vision (BMCV'02), Tuebingen, Germany,* in-press.
12. G. M. Nelson and R. D. Quinn. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 1, pages 157–162, 1998.
13. R. A. Rensink. *Vision Res*, 40(10-12):1469–1487, 2000.
14. M Riesenhuber and T Poggio. *Nat Neurosci*, 2(11):1019–1025, Nov 1999.
15. B. Webb. *Behavioral and Brain Sciences*, 24(6), 2001.
16. B Werger and M J Mataric. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):173–198, 2001.