Visually-Guided Adaptive Robot (ViGuAR)

Gennady Livitz, Heather Ames, Ben Chandler, Anatoli Gorchetchnikov, Jasmin Léveillé, Zlatko Vasilkoski, Massimiliano Versace, Ennio Mingolla, Greg Snider, Rick Amerson, Dick Carter, Hisham Abdalla, and Muhammad Shakeel Qureshi

***Abstract*— A neural modeling platform known as *Cog ex Machina*[1] (Cog) developed in the context of the DARPA SyNAPSE[2] program offers a computational environment that promises, in a foreseeable future, the creation of adaptive whole-brain systems subserving complex behavioral functions in virtual and robotic agents. Cog is designed to operate on low-powered, extremely storage-dense memristive hardware[3] that would support massively-parallel, scalable computations.**

We report an adaptive robotic agent, ViGuAR[4], that we developed as a neural model implemented on the Cog platform. The neuromorphic architecture of the ViGuAR brain is designed to support visually-guided navigation and learning, which in combination with the path-planning, memory-driven navigation agent – MoNETA[5] – also developed at the Neuromorphics Lab at Boston University, should effectively account for a wide range of key features in rodents' navigational behavior.

I. Introduction

THE ability to search the environment for features consistent with behavioral objectives, to approach a selected target, or to avoid an identified obstacle is critical for an animal's survival. The same abilities are equally important for robots, since visual search and visually-guided navigation are the essential elements for most of the tasks robots perform. Animals are highly effective in performing navigational tasks in unknown environments; they are capable of orienting using landmarks, path-planning, approaching objects they like or avoiding objects they have already visited, depending on their behavioral objectives[6]. All these tasks are very relevant to robots, and therefore the principles guiding navigational behavior in animals as well as the neural architecture of circuits involved in navigation are of great interest to neuromorphic engineering.

Visually-guided search and place-recognition triggered response are the first two types in a hierarchical typology of navigation behavior[7]. Both these types of navigation are used when an object can be perceived, but the latter requires some form of cognitive map being employed to represent an association between the object and location to be learned by an animal.

The purpose of this project was to model these types of navigation on the Cog[1] platform. Cog is designed to operate on low-powered, extremely storage-dense memristive hardware; however its hardware abstraction layer makes development on the Cog platform independent from an underling hardware platform and thus suitable for development robotic applications that are based on GPU or any other computational platform currently available.

Cog is a distributed, massively-parallel software platform for running neuromorphic applications. Cog uses tensor transformation concepts to describe processing that takes place in computational nodes, where each thread transforms multiple input tensors into a single output vector (fiber). Cog adopts a completely deterministic model of data processing that enforces globally synchronous computations. Cog is being successfully used to model learning and navigation behavior in virtual environments[5]. However, the ability to use sensory information, to learn, and to deal with physical properties of the real world creates unique challenges for both modeling algorithms involved in navigation and modeling environments. Addressing these challenges is the objective of the ViGuAR project.

II. Components of the ViGuAR Brain Architecture

The architecture shown in Fig. 1 constitutes a robotic brain of an iRobot Create that learns to associate the color of an object with a reward while living in a world populated by red and green cylindrical objects of fixed size (Fig. 2).

The robot receives its visual input from a netbook's webcam. The netbook is aligned with the robot's head direction and connected to the robot via a serial port. The serial port interface is used to send motor commands and to receive sensory (touch) events from the robot's front bumpers.

Figure 1 shows the data flow that transforms the sensory information (visual and touch) into motor behavior (Motor Output), which results in the robot approaching or avoiding an object. A brief description of functional components of the ViGuAR brain architecture follows.

The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.Manuscript received February 10, 2011. This work was partially funded in part by the DARPA SyNAPSE program, contract HR0011-09-3-0001. Massimiliano Versace, Ennio Mingolla, Heather Ames, and Ben Chandler were also supported in part by CELEST, a National Science Foundation Science of Learning Center (NSF SBE-0354378 and NSF OMA-0835976).

G. L. (corresponding author) is with the Department of Cognitive and Neural Systems, Boston University, Boston, MA 02215(e-mail: glivitz@gmail.com).

Z. V. is with Harvard Medical School.

H. A., B.C., A.G., J. L. M. V., and E. M. are with the Department of Cognitive and Neural Systems, Boston University, Boston, MA 02215.

G. S., R. A., D.C., H. A., and M. S. Q. are with Hewlett-Packard Laboratories.

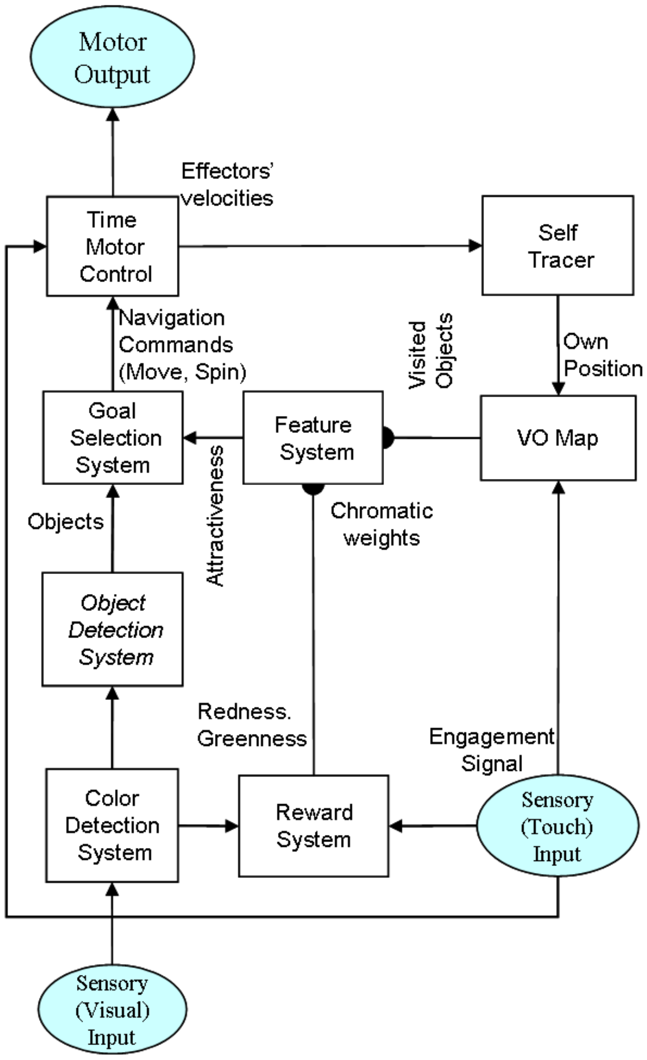


Fig. 1. ViGuAR Brain Architecture
The arrows show the direction of the data flow. The round bold heads (adaptive connection) designate data flow that is based on persistent long-term memory stored at the module in which connection originates.

The Color Detection System (CDS) converts an *RGB* tensor produced by the webcam into a tensor that represents chromatic features: *redness* and *greenness*.

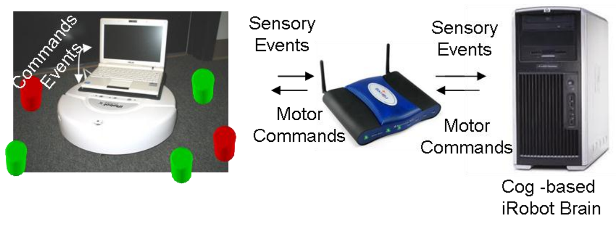
The Object Detection System (ODS) converts the color tensor into a binarized object representation. In a simplified version of the ViGuAR brain, a discontinuity in color features is recognized as an object's boundaries. Positions of objects and their dimensions are used by the Goal Selection System (see below) to produce motor commands.

The Reward System (RS) produces a positive or negative reinforcement by associating chromatic features of the contacted object with a corresponding reward value. The reward value acts as a teaching signal, which may be related to any property of the object. Based on input from the bumper sensors, weight adjustment occurs when the robot collides with an object.

The Feature System (FS) computes a tensor of *attractiveness* based on *weights* associated with the chromatic features: redness and greenness. The attractiveness tensor is convolved with a *projection of visited objects* in the robot's current view, and thus suppresses the attractiveness for locations that ViGuAR remembers as belonging to objects it has approached in the past.

.

The Goal Selection System (GSS) analyzes the



attractiveness tensor in order to find the most attractive goal in its view. It uses the objects' representation to evaluate the distance to a target. GSS integrates *attractiveness* data produced by the FS from multiple views in order to select an optimal goal and generate a proper motor command. The goal selection is based on past reward history.

The Robot Tracer (RT) updates the robot's vector of movement upon completion of each motor command. Thus, location of the robot is constantly traced in a short term memory with respect to the robot's initial position.

The Visited Objects Map (**VO Map**) contains a representation of space where the robot operates. Locations that belong to objects that the robot has previously approached are set to a value that denotes "visited". The VO Map is populated based on a signal from the RT when contact with an object occurs. The VO Map signal is used by the FS to produce the robot's current view projection in order to prevent revisiting objects that the robot has visited in the past.

Fig. 2. ViGuAR Visual Environment
The robot operates in the world of red and green cylindrical objects

III. Communication Model

The architecture shown in Fig. 2 resides a Cog-based brain on a workstation that communicates with the netbook attached to the robot's serial port over a wireless network.

A server uses the UDP protocol to receive motor commands and sensory events and dispatches them to the registered robotic clients. Immediate communication with the robot is controlled by Python clients running on a Linux Ubuntu OS.

While easing performance constraints related to housing a brain on a netbook, this communication paradigm also creates additional challenges related to remote and potentially unreliable access to sensory information and the robot's effectors.

IV. ViGuAR's Visual Search and Navigation Strategy

The robot essentially operates in a flat world, in which the visual representation involved in navigation decisions is reduced to a single dimension, namely azimuth. The strategy used by ViGuAR to collect the visual input given the limited viewing angle of the webcam is to spin half of the visual view angle (Fig. 3) in order to catch an object in the middle of its view in case it falls on the view periphery. Using overlapping views helps to minimize errors related to seeing objects partially on a viewing edge. ViGuAR performs a search for an attractive object by analyzing attractiveness in a particular viewing direction. This is done independently from an objects' boundary determination. Once an attractive location is found, the robot orients towards it and integrates attractiveness with object boundaries as determined by the Object Detection System. This allows the Goal Selection System to determine the distance that the robot needs to travel to approach a target.

As the robot seeks to navigate in a horizontal plane, the ViGuAR brain reduces a two-dimensional neural representation to a single dimension (Fig. 4) centered on the head direction in a robocentric coordinate frame. Before approaching an object, the robot orients itself toward the middle of a target.

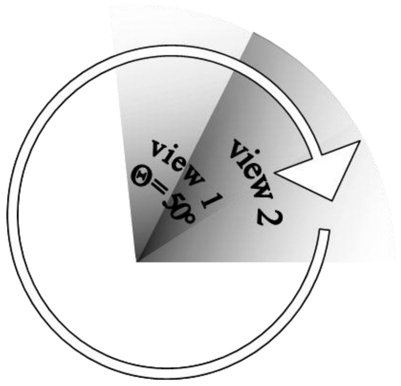


Fig. 3. Overlapping Visual Views
The robot rotates ½Θ to minimize errors in distance evaluation

To approach a target, the robot iteratively performs the following sequence:

1) Determine if there are obstacles (other objects) on its way to a target (position 1, Fig. 4).
2) If the pathway of the robot to the target is not obstructed, move straight to the target.
3) If the trajectory is obstructed (Fig. 4) spin to avoid collision with the closest obstacle and move to the closest obstacle (position 2 on Fig. 4).
4) Perform a visual search procedure to find an unobstructed way to the target (position 3, Fig. 4).

Thus, the robot's object approaching trajectory is formed by a chain of avoidances of the next closest object.

Having two independent systems providing input to the GSS results in an effective computational solution in which the goal-orienting behavior is achieved based just on attractiveness supplied by the Feature System, while the Object Detection System, which works in parallel, assumes that the robot is oriented toward the goal. This assumption is reasonable because until the robot is oriented toward the goal, the signal from the ODS is ignored by the GSS.

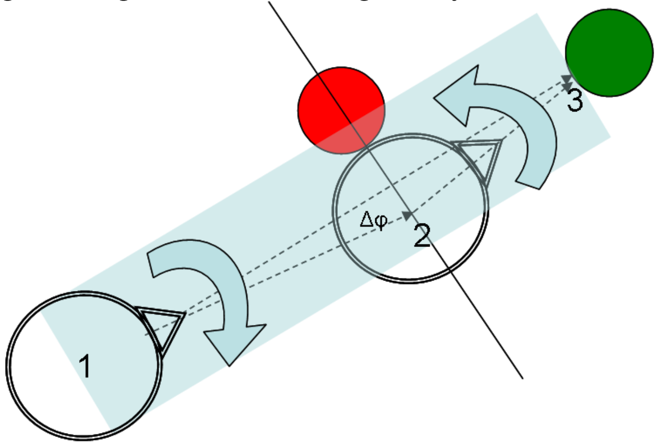


Fig. 4. Object Approaching Strategy
The robot rotates clockwise to avoid collision with the red object and then rotates counterclockwise to reach the green object.

Orienting toward an attractive location can be compared with the deployment of attentional resources in biological visual systems. At the same time, some locations are primed as not worthy to attend to via the object map projection, which blocks visual signals from locations that the robot has already visited.

ViGuAR maintains persistent object representation in a two-dimensional model of space – the VO Map. An active entry in this map corresponds to a spatial location that is learned as belonging to an object. This map is projected onto a one dimensional "retinal" view at the robot's current position in order to be convolved with the attractiveness data in the direction of the robot's orientation.

ViGuAR uses a reinforcement learning paradigm[8]. Learning occurs when the robot collides with an object. On contact, the weights representing color features forming color attractiveness are modified according to an external reward signal. The reward signal changes the weights associated with the color features of an object approached by the robot. The weights could change in either a positive or negative direction.

V. Cog Implementation of the Viguar Brain

The architecture in Fig. 1 is implemented as a Cog brain model (Fig. 5). The network illustrated in Fig. 5 is produced by the Cog visualization subsystem as a reflection of connections specified by a program written in the SCALA language using the Cog 2.0 API. The output of each node in this network is a tensor sent along outgoing connections. Multiple incoming connections are treated as multiple incoming tensors. Each node's processing unit defines processing that produces a single fiber in an outgoing tensor. Processing in all nodes takes place in parallel and completes within a single Cog cycle thus ensuring computational determinism.

Two independent pathways produce object boundaries and color attractiveness information that gets integrated by the Goal Selection System. The Memory System along with the Sensory and Reward System form a feedback mechanism that contributes to navigational decisions and thus influences motor behavior of the robot.

The diagram in Fig. 5 has 500364 neural elements, which make 658244 connections, 40002 of which represent persistent modifiable elements that function as long term memory.

A. *Sensory Input*

A collection of visual data is performed by the webcam that functions as a robotic retina - a receptoral level of visual processing. The webcam attached to the robot has a viewing angle of about 50º. The webcam produces a two-dimensional RGB matrix. The resolution used by the ViGuAR brain "retina" is H by V (320 X 240 pixels). Thus, the output from the module that collects raw visual input is a 3D-tensor encoding R, G, and B intensity for each "retinal" location.

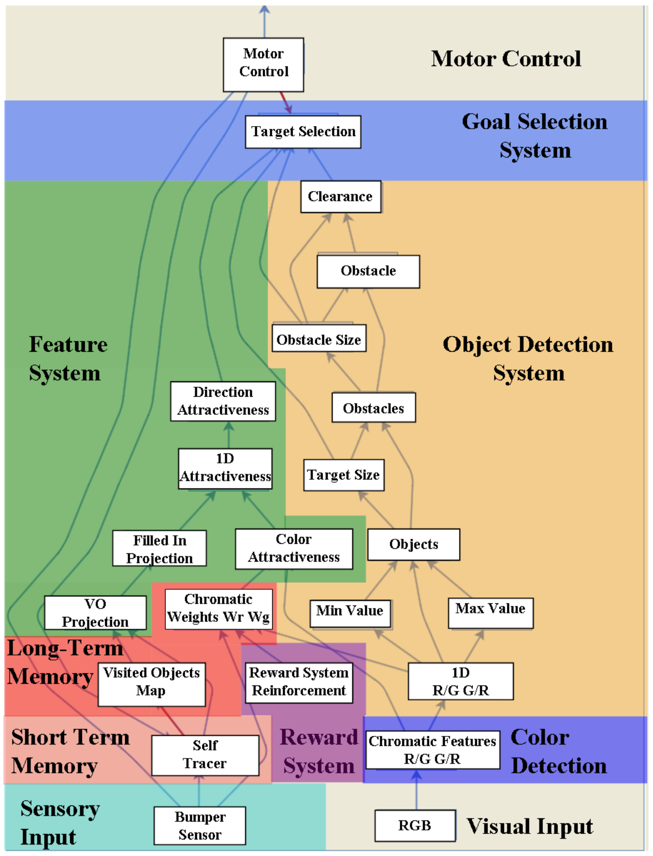


Fig. 5. Viguar's Cog-based Neural Network
Color encodes functional components of the ViGuAR brain architecture.

Sensory information for touch comes from the robot's bumpers. ViGuAR does not distinguish between the left and the right bumper signals and delivers a "bumper touched" event when either of two bumpers is engaged with an object.

B. *Color Detection*

Color Detection[9] produces color features: redness R_G and greenness G_R out of the RGB tensor.

$$R_G = \frac{[R-G]^+}{R+G}; \quad (1)$$

$$G_R = \frac{[G-R]^+}{R+G}. \quad (2)$$

The above formulas give a rough approximation of chromatic features sufficient for color detection and object identification in the world of red and green objects. The output from the Color Detection System is sent up to the Object Detection System and the Feature System for boundary detection and attractiveness evaluation.

C. *Object Detection*

Since ViGuAR operates in a world populated by flat objects, the 2-D output from the Color Detection System is first squeezed in the vertical direction by averaging the value of color features R_G and G_R across a vertical dimension; this produces a 1-D signal for redness $\boldsymbol{r_g}$ and greenness $\boldsymbol{g_r}$ respectively (output from the unit 1D R/G G/R module).

The next cascade of processors transforms r_g and g_r one-dimensional signals into the binary representation of objects R_N and G_N, by normalizing the input between minimum and maximum values of $\mathrm{r_g}$ and $\mathrm{g_r}$ and applying thresholds $\mathrm{T_R}$ and $\mathrm{T_G}$ according to (4) and (5).

$$r_g = \frac{\sum_H R_{G_i}}{H} ; g_r = \frac{\sum_H G_{R_i}}{H} ; \qquad (3)$$

$$R_N = \left[\frac{\mathrm{r_g} - r_{\min}}{\mathrm{r_{max}} + \mathrm{r_{min}}} - T_R \right]^+ ; \qquad (4)$$

$$G_N = \left[\frac{\mathrm{g_r} - \mathrm{g_{min}}}{\mathrm{g_{min}} + \mathrm{g_{max}}} - T_G \right]^+ . \qquad (5)$$

This representation binarizes all detected objects that are in view. An object may become a target to be achieved or an obstacle to be avoided.

To support a goal reaching strategy, the ODS should segment the visual image, in order to produce boundaries and determine the size of both the target and the closest obstacle. The ODS assumes that the robot is oriented towards a target; therefore it determines the size of the object in the center of the robot's view and removes the central object from the obstacles' projection. This operation is performed by the Target module (Fig. 5). The output from the Obstacles unit contains projection of all obstacles to the robot's 1-D retinomorphic representation. Finally, the ODS finds the closest obstacle (the one with the largest projection) and deletes all other obstacles from the 1-D projection. The final output from the Obstacles module contains just one projection – a projection of the closest obstacle.

This 1-D projection is convolved with self-projection as it reaches the obstacle on its way to the target (Fig. 4). A zero result of this convolution (output from the Clearance module) means that the robot can avoid collision with an obstacle and therefore can start moving toward the target.

D. *The Feature System*

The output of the Feature System is a 1-D directional projection of attractiveness. Projection of attractiveness is produced by gating the color attractiveness computed from chromatic feature signals, (redness and greenness) by a one-dimensional projection of the visited object memory in the robot's current view. Transformation of chromatic features into attractiveness undergoes the following steps:

1) Chromatic features are transformed into color attractiveness *A* by weighting chromatic signals using the weights $\mathrm{W_R}$ and $\mathrm{W_G}$ associated with redness $\mathrm{r_g}$ and greenness $\mathrm{g_r}$ respectively.

$$A = W_R r_g + W_G g_r . \qquad (6)$$

2) Color attractiveness *A* is reduced to one dimension and gated by 1D-projection of the visited object map *O*. *O* is set to -1 when it represents a projection of a location that belonged to an object visited by the robot. 1D measure of attractiveness *a* is produced according to (7).

$$a = \frac{1}{V} \sum_V A_i * O, \qquad (7)$$

where *V* is a vertical dimension of the visual input matrix *H* by *V*.

a is convolved with a rectangular kernel in order to produce Direction Attractiveness A_D, which peaks at locations in the center of color-attractive objects. Closer objects that result in larger projections produce larger signals and thus, have better chances to attract the Goal Reaching System. Direction Attractiveness is defined as:

$$A_D = \sum_{i-K \text{ to } i+K} a_i \quad , \qquad (8)$$

where K is the size of the kernel (K=10)

E. *The Reward System*

The output from the reward system weighs chromatic signals that produce color attractiveness. A reward value is produced when the robot hits an object. Based on the external signal that could represent a temperature or taste in a real world environment, the Reward System produces positive or negative reinforcement for the chromatic features (redness or greenness) associated with the object that the robot just contacted.

Learning obeys the following law:

$\mathrm{W_X(i) = W_X(i) + \Delta W_X}$, where

$$\Delta W_X = \left| \begin{array}{l} X * \Delta^- * E \\ (1 - W_X) * \Delta^+ * E, \end{array} \right. \qquad (9)$$

where *X* – represents chromatic signals redness R_G and greenness G_R, W_X – the weight associated with a chromatic feature X, *E* is an external teaching signal that can be either 1 or -1, Δ^+ is learning rate of positive reinforcement in the range [0, 1] (we used $\Delta^+ = .5$); Δ^- is learning rate of negative reinforcement in the range [0 1] (we used $\Delta^- = .9$).

If the initial values for W_x are set in the range [0, 1], the learning law (9) would contain the weight values in the range from 0 to 1.

F. *The Goal Selection System*

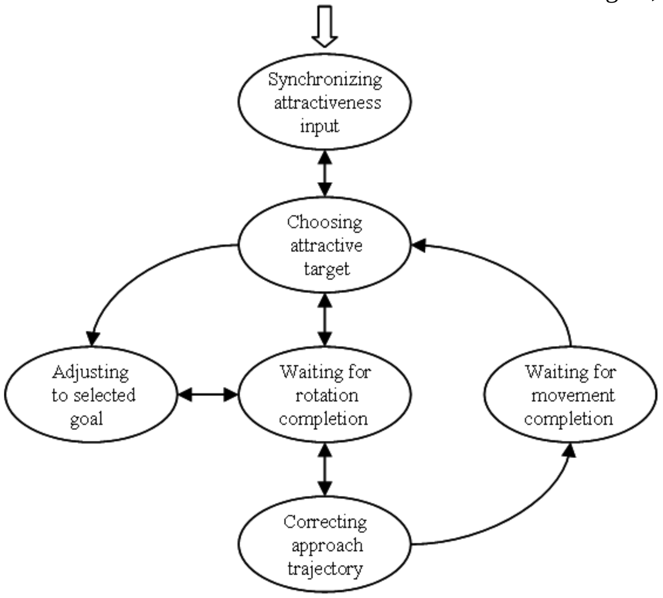
The goal of the Goal Selection System is to initiate motor behavior in order to fulfill the robot's attraction to certain colors. The analysis of environment in search for an attractive goal goes through the following phases:

1) Analysis of attractiveness at robot's current orientation,
2) Analysis of attractiveness at robot's current position,
3) Analysis of the selected target.

Fig. 6. Goal Selection System State Diagram

These phases are reflected in the Goal Selection System state diagram (Fig. 6). Collection of visual data at the robot's current position involves acquiring attractiveness data until the attractiveness at some direction amounts to the level required to initiate movement. This level is specified by the *urgent movement threshold T_U.* Due to a limited viewing angle, collection of data at the current spatial position involves multiple spins followed by visual data acquisition and subsequent analysis of a direction's attractiveness. When the direction's attractiveness exceeds the urgent movement threshold $T_{U,}$ the Goal Selection System initiates movement toward the most attractive position.

If all data that the robot can see from its position are collected and the attractiveness in any direction does not exceed the urgent movement threshold, the Goal Selection System chooses the most attractive direction in the entire 360 degree world and initiates rotation toward it if its attractiveness exceeds the second movement threshold $T_{M.}$ If none of the directions merits a movement toward the goal,


Synchronizing attractiveness input
Choosing attractive target
Adjusting to selected goal
Waiting for rotation completion
Waiting for movement completion
Correcting approach trajectory

the Goal Selection System gives up goal selection at its current position and initiates the robot's movement to a new one.

The Goal Selection System does not store visual view data. Instead, it keeps track of the maximum attractiveness and its direction with respect to the robot's orientation at which the visual search began. Thus, the current view data are compared with the stored maximum, which can be updated as a result of this comparison.

The angular position of the maximum attractiveness relative to the robot's initial orientation can be computed as

$$\varphi_{\max} = .5H * R_{\max} + P_{\max} \quad , \tag{10}$$

where R_{max} is a number of rotations made by the robot before it detected the maximum, P_{max} is the location of the maximum relative to the view where it was detected, and H is the size of the one-dimensional retinal projection (320 pixels).

Current orientation of the robot relative to its initial orientation can be computed as

$$\varphi_{current} = .5H * R_{current} + P_{current} \; . \tag{11}$$

Thus, the angle that the robot needs to spin from its current orientation toward the maximum is

$$\Delta\varphi = \frac{(\varphi_{current} - \varphi_{\max})}{H}\Theta \quad , \tag{12}$$

where Θ is the size of robot's visual angle - 50°.

Due to accumulation of error related to imprecise execution of motor command by the robot, an actual angle that the robot rotated toward the direction associated with maximal attractiveness may be different from the one computed by the above formulas (10-12), and therefore the robot, after completion of phase 2, might become misaligned with the direction of maximum attractiveness. To overcome this error, before initiating the movement toward the anticipated goal, the Goal Selection System adjusts the orientation of the robot toward the location of maximum attractiveness within its current view. This adjustment is computed as:

$$\Delta\varphi_{adj} = \frac{P_{\max} - .5H}{H}\Theta \tag{13}$$

Once the robot is oriented towards the goal, the GSS checks if the robot's straight path to the goal is unobstructed. It uses a signal from the Object Detected System to make sure that this signal is zero. Having a non-zero signal from the Clearance module means that the robot would hit an obstacle on its straight path to the target. In this case, the angular shift $\Delta\varphi$ in robot orientation required to clear the path toward the target (Fig. 3) can be computed as shown in Fig. 7.

X_1 and X_2 points (Fig. 7) are projections of the robot's own position on its straight pass to the target and robot's own position on its corrected path that avoid collision with the obstacle (Fig. 4). These positions are essentially the beginning and the end of the projection of overlap computed by the Clearance module in the Object Detection System.

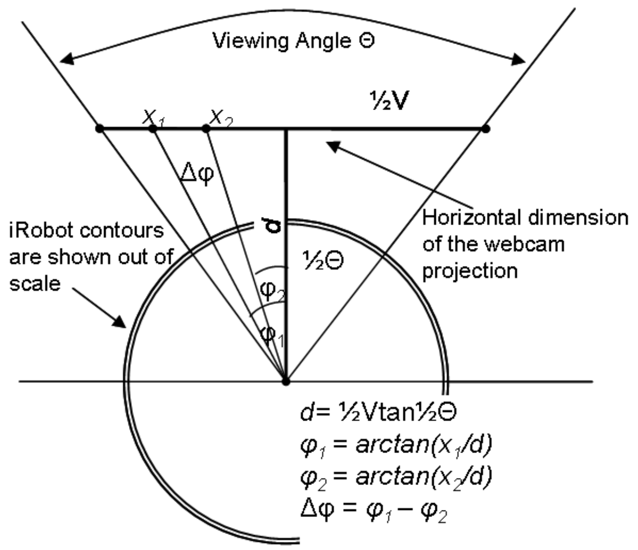

Viewing Angle Θ
x_1
x_2
½V
Δφ
Horizontal dimension of the webcam projection
iRobot contours are shown out of scale
d
½Θ
φ_2
φ_1
d= ½Vtan½Θ
$\varphi_1 = arctan(x_1/d)$
$\varphi_2 = arctan(x_2/d)$
$\Delta\varphi = \varphi_1 - \varphi_2$

Fig. 7. Computing the Angle Required to Avoid Obstacle

When the robot is finally oriented toward a selected goal (Fig. 8), the Goal Selection System evaluates the distance to the target. Knowing the size of the target's 1-D projection, the distance *D* to the target can be evaluated as

$$D = \frac{Od}{o}. \quad (14)$$

The initial distance evaluation is made only on the basis of visual input; however that distance gets readjusted every time the robot passes by an obstacle (Fig. 4) or collides with an object. Such readjustments are reflected by the Self Tracer module and eventually make their way in the VO Map. These distance readjustments are made based on the distances actually traveled by the robot computed on the basis of time, wheel speed, and sensory information. In the future we plan to add proprioception ("spinal cord") to the robot brain model to make its self-tracing more accurate.

G. Motor Control

The output from the Motor Control module specifies the activity sent to robot effectors. A two-value vector defines the velocity of the robot's left and right wheels. The robot moves when both components of the vector have the same sign and spins when the signs are opposite. Positive values set for both wheels make the robot move forward, while negative values make the robot back up. Clockwise and counter-clockwise rotations are specified by setting the first element of the movement vector to a positive and negative value, respectively.

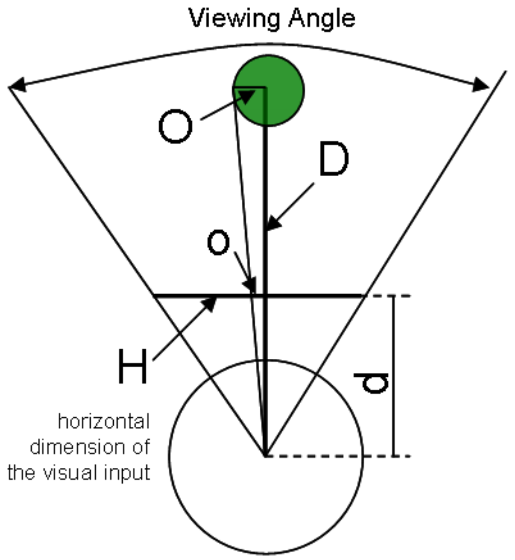


Fig. 8. Computing Distance to the Target

The Motor Control module controls the timing of motor activity required to achieve a motor objective produced by the Goal Selection System. It bases its timing estimates on ViGuAR brain performance, robot wheel speed, and a motor objective, which could either be an angle to be rotated T_A or a distance to be traveled T_D, computed as:

$$T_A = \frac{\varphi R}{f\omega}; \quad (15)$$

$$T_D = \frac{D}{f}, \quad (16)$$

where φ is an angle to be rotated, *R* is iRobot rotation radius (which corresponds to the distance from the robot's center to its wheel), *f* is the robots wheel speed, ω is the ViGuAR brain performance in computational cycles per second, and D is the distance to be travelled as determined in Fig. 8. The formulas (15-16) yield time as a number of Cog cycles required to complete a given movement.

If the outgoing activity of the Motor Control module is not zero, the robot produces corresponding motor behavior (moving or rotating) unless it is engaged with an object. Being engaged with an object blocks execution of the motor command that led to the engagement until the output from the Motor Control module changes.

The Motor Control module processes sensory input from the bumpers. Presence of the sensory input signals a contact with the object in the robot's head direction. When such an event occurs, the Motor Control module reverses the robot's current motor activity. The Motor Control module's outgoing activity is also sent to the Robot's Tracer – a module that traces position of the robot with respect to its initial position.

H. Memory System

The memory system traces the position of the robot and places locations of objects on a map it dynamically constructs while the robot explores the environment. The main function of the memory system is to produce projections of objects that the robot has already visited on the robot's 1-D directional view. This projection gates the attractiveness signal from the Feature System, thereby making locations that the robot has already visited as not attractive. The memory system consists of two major components: the robot's tracer and its memory map, which act as the short and long term memory respectively.

The Robot's Tracer receives its input from the Motor Control unit and updates the robot's position once the Motor Control output changes. Each of the robot's movements can be described by its angular and modular components, which are updated separately due to the fact that the robot either rotates or moves straight. Thus, the position of the robot can be defined as the sum of two vectors: a vector defining the robot's current position and a vector that defines the robot's movement:

$$\vec{P}_{new} = \vec{P}_{current} + \vec{M} \, . \tag{17}$$

The vector M components (φ, ρ) can be found as

$$\varphi = \frac{tf}{R\omega} \; ; \tag{18}$$

$$\rho = \frac{tf}{\omega} \, , \tag{19}$$

where t is the time of the motion (rotation or forward/ backward movement) in Cog cycles.

The robot's Cartesian increments Δx and Δy in 2-D space can be found as:

$$\Delta x = \rho \cos \varphi, \text{ and} \tag{20}$$

$$\Delta y = \rho \sin \varphi. \tag{21}$$

And the new robot's coordinates X_r and Y_r are

$$X_r = x_r + \Delta x, \text{ and} \tag{22}$$

$$Y_r = y_r + \Delta y, \tag{23}$$

where x_r and y_r are robot's current coordinates.

The signal from the robot's bumpers triggers the process of placing an object at the robot's current location on the memory map. The coordinates of the contacted object can be found on the trajectory of the robot at the distance that combines the length of the radius between the robot and the object. Thus, the center of the object that the robot hit can be found as

$$X_o = X_r + (R_r + R_o) \cos \varphi, \text{ and} \tag{24}$$

$$Y_o = Y_r + (R_r + R_o) \sin \varphi, \tag{25}$$

where X_o and Y_o are object coordinates and R_r and R_o are the robot and object radii respectively.

The actual unit that implements the long term memory remembers locations where the robot hit an object. It populates a memory map based on the signal from the robot's tracer by setting locations within a circle of radius R_o from the center of object locations as computed by the robot's tracer.

$$F(x,y) = \begin{vmatrix} 1 & (x - X_0)^2 + (y - Y_0)^2 \le R_o^2 \\ 0 & (x - X_0)^2 + (y - Y_0)^2 > R_o^2 \end{vmatrix} \tag{26}$$

The resolution of the VO Map ultimately determines the accuracy of navigation with respect to avoidance of revisiting the same object. It is determined by two parameters: the size of the VO Map (in pixels) l and the size of the physical space remembered by the ViGuAR L. The ratio of l/L defines the VO Map resolution – the area represented by one pixel.

I. Convolving the VO MAP with the Current Visual Input

The output from the Visited Objects' map F is always projected onto the robot's current view for gating the real time visual signals, thus preventing the robot from revisiting a spatial location mapped as belonging to an object.

Producing a 1-D projection of the visited object map is performed by the VO Projection unit. In order to fall within a robot's view, the location (x, y) should obey the following inequality:

$$|arctg\,(y/x)| < \tfrac{1}{2}\Theta \, . \tag{27}$$

The above inequality assumes direction of the axis X aligned with the robot's view and location of (0, 0) aligned with the robot's position. In order to verify the above inequality, the VO Projection unit performs a linear transformation effectively transitioning from the world coordinates aligned with the robot's initial position to the robot's egocentric coordinates aligned with the robot's view defined as:

$$x = (x_o - X_R) \cos \varphi - (y_o - Y_R) \sin \varphi, \text{ and} \tag{28}$$

$$y = (x_o - X_R) \sin \varphi + (y_o - Y_R) \cos \varphi \tag{29}$$

where X_R and Y_R are robot coordinates in the world coordinate system, x_o and y_o are object coordinates in the world coordinate system, and x and y are coordinates in the robot's egocentric coordinate system.

The actual position of the point P_R in 1-D projection can be found as:

$$P_R = d\,\frac{y}{x} + .5H \tag{30}$$

The location of the robot is aligned with the middle of 1-D projection. Finally a limited resolution of the space representation in the object map requires filling in the gaps in 1-D projection so the objects appear continuous. This is performed by the Filling-in projection module. The output from this module is convolved with the Feature System output to produce attractiveness used by the Goal Selection System.

VI. CONCLUSION

Video of ViGuAR simulations[10] demonstrate that Cog is a viable platform for neuromorphic engineering. It enforces massive parallelism of neural computations and tensor data representation resulting in solutions that bear similarities with biological brains[11], such as synaptic plasticity, layered architecture, parallel processing pathways (such as “what” and “where” pathways), among others. Eventually we plan to integrate our virtual environment agent MoNETA and our robotic agent ViGuAR to broaden the range of behaviors and speed up the development of more advanced capabilities, including a visually-guided adaptive navigation. Future development of the ViGuAR model will include the ability to operate in progressively more sophisticated visual environments that include a variety of visual features such as shape, color, and depth, among others. Additionally, ViGuAR's future adaptive behavior repertoire will include learning various visual features and using a variety of biologically plausible learning laws with a final goal of replicating classic experimental results from behavioral neuroscience literature of rodents.

REFERENCES

[1] G. Snider, R. Amerson, D. Carter, H. Abdalla, S. Qureshi, J. Léveillé, M. Versace, H. Ames, S. Patrick, B. Chandler, A. Gorchetchnikov, and E. Mingolla, “Adaptive Computation with Memristive Memory”, *IEEE Computer*, February v. 44 n. 2, pp. 46-53, (2011)

[2] The SyNAPSE Project http://cns.bu.edu/nl/synapse.html.

[3] D. B. Strukov, G. S. Snider, D. R. Stewart, R. S. Williams, “The missing memristor found”, *Nature* 453, pp. 80-83 (2008).

[4] G. Livitz, H. Ames, B. Chandler, A. Gorchetchnikov, J. Léveillé, Z. Vasilkoski, M. Versace, E. Mingolla, “Scalable adaptive brain-like systems”, *Neuromprphic Engineer*, to be published.

[5] M. Versace, B. Chandler, “The Brain of a New Machine”, *IEEE spectrum*, v.47, n.12, pp.30-37 (2010).

[6] R. D'Hooge, P. P. De Deyn, “Applications of the Morris water maze in the study of learning and memory”, *Brain Research Reviews*, Volume 36, Issue 1, August 2001, pp 60-90, (2001).

[7] O. Trullier, S. I. Wiener, A. Berthoz, .A Meyer, Biologically Based Artificial Navigation Systems: Review and Prospects, *Progress in Neurobiology*, 51, April 5, pp. 483-544 (1997).

[8] R. S. Sutton, A. G. Barto, “Reinforcement Learning: An Introduction*”*. MIT Press (1998).

[9] Eskew R. T., McLellan J. S.,Giulianini F. in *Color Vision: From Genes to Perception*, eds Gegenfurtner K R , Sharpe L T (Cambridge Univ. Press,New York), pp. 345–368 (1999).

[10] Neuromorphics Laboratory, Robotic Simulations http://cns.bu.edu/nl/robotics.html

[11] T. J. Prescott, P. Redgrave, and K. Gurney, “Layered Control Architectures in Robots and Vertebrates” *Adaptive Behavior* January 7, pp.99-127(1999).