TECHNICAL REPORT NO. 378

Lukasiewicz' Insect: The Role of Continuous-Valued Logic in a Mobile Robot's Sensors, Control, and Locomotion

By

Jonathan W. Mills

March 1993

COMPUTER SCIENCE DEPARTMENT

INDIANA UNIVERSITY

Bloomington, Indiana 47405-4101

Łukasiewicz' Insect: The Role of Continuous-Valued Logic in a Mobile Robot's Sensors, Control, and Locomotion

Jonathan W. Mills

Indiana University
Bloomington, Indiana 47405

Abstract

The ability to physically realize a colony of insect-like robots presents numerous problems to robotics researchers. A hexapod robot controlled by a computational sensor is proposed as a solution to some of these problems. Stiquito is a small nitinol-propelled robot. It is controlled by a computational sensor implemented with Łukasiewicz logic arrays (ŁLAs). The computational sensor includes an electronic retina, an implicit controller, and a gait generator. Measured and simulated results illustrate the unifying effect of Łukasiewicz logic on the design of the robotic system.

Keywords: *analog VLSI*
Łukasiewicz logic array
robotics
subsumption architecture
Stiquito

1: Introduction

1.1: The complex insect

Research in robotics has advanced recently by lowering its expectations. Problems in path-planning, modelling, and image understanding were the focus of robotics research from 1960 to 1986. These problems are difficult in any domain, and have yet to be solved satisfactorily.

In 1986 Brooks proposed studying simpler robotic systems controlled by behavior-oriented rather than function-oriented networks. This approach is known as a *subsumption architecture* [1]. Attila and Genghis, two robots controlled by a subsumption architecture, resemble insects yet are capable of learning complex behaviors such as walking [2]. The concept of subsumption architecture has been extended to model the complete behavior of a cockroach-like robot [3], and to the study of insect-like robot colonies [4]. But implementing a single insect-like robot is still difficult, and robot colonies are even more so. To understand why, let's consider the biological competition: the ant.

Ants are sophisticated creatures. They have eyes that implement lenses and filters using binary optics; they communicate both by touch and chemically using pheromones; they are strong for their size (capable by analogy of climbing and descending Mount Everest five times a day while carrying their own weight) [5]; and they have multiple degrees of freedom in each leg, antenna, and the mandibles. By contrast insect-like robots are clumsy and ponderous. Their optics range from simple photocells to television cameras; they cannot communicate using touch or chemicals; they are fragile and weak for their size; and they have only a few degrees of freedom in each leg or arm.

A single ant colony may contain up to 10,000 adults ranging in size from 3mm to 25mm in length. A colony has many castes specialized for specific tasks, yet capable of acting in concert to perform complex actions (bridging, wars, agriculture). Colonies of up to 20 robots have been implemented with "adults" approximately 500mm in length that cost $2000 each. Of the robot colonies physically realized to date none have castes designed for specific tasks, and concerted action is simple (such as flocking behavior). The weight-to-power ratio of small self-contained robots limits the functionality of their control and locomotion systems, which even then are difficult to construct.

The use of continuous-valued logic to implement a simple subsumption architecture for the small robot, *Stiquito,* is proposed as a step toward the solution of these problems.

1.2: Stiquito: Łukasiewicz' insect

Stiquito is an insect-like hexapod robot propelled by nitinol actuator wires [6]. It is small (60mm long × 70mm wide × 25mm high), lightweight (10 gm), and inexpensive ($10). Stiquito is capable of carrying up to 200 grams while walking at a speed of 3 to 10 centimeters per minute over slightly textured surfaces. Its payload typically consists of sensors, control and drive electronics, and a 9-volt cell. The robot walks when heat-activated nitinol actuator wires attached to the legs contract. The heat is generated by passing an electric current through the nitinol wire. The legs can be actuated individually or in groups to yield tripod, pacing, and other gaits.

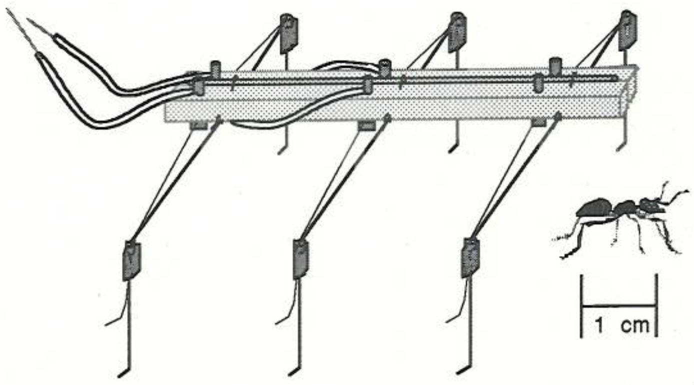


Figure 1. Stiquito and an ant (both ×1)

1.3: Robotics and continuous-valued logic

It is a paradox of robotics research that inherently imprecise and inaccurate biological systems are often implemented with digital systems far more precise and accurate than the original.

Digital implementations of hierarchical systems are flexible and easily programmable. However they weigh too much and consume too much power to be used to control an autonomous Stiquito. These systems are also unnecessarily complex. The programmability of digital microprocessors is useful during prototyping, but unnecessary in the final implementation; programmability can be emulated during design by simulation of the system. Conversion of analog sensor inputs to a digital representation and back to analog control outputs adds additional complexity that is not needed by an imprecise insect-like robot.

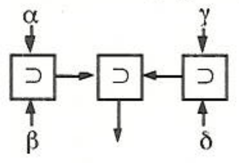
Consequently, a fully analog system was chosen to obtain lightweight, low-power, and simple sensors, controllers, and actuators, avoiding the paradox. Łukasiewicz logic arrays are used throughout the system. The continuous-valued Łukasiewicz logic provides a unified framework for the design of Stiquito's sensors, controllers, and locomotion.

1.4: Łukasiewicz logic arrays

Łukasiewicz logic is a multiple-valued logic with a denumerably infinite number of truth values [7]. Real circuits are described by subsets of the logic that have a finite number of continuously-varying truth values. No circuitry is required to quantize the logical values in the circuit. The limit to the number of values that can be encoded on a wire occurs when values cannot be distinguished because of noise [8].

Łukasiewicz implication is defined by the valuation function $v(\alpha \supset \beta) = min(1, 1-\alpha+\beta)$. *Negated implication* has a valuation function defined as $v(\alpha \not\supset \beta) = max(0, \alpha-\beta)$. The term *implication* is used to refer to both functions.

Łukasiewicz logic arrays (ŁLAs) are arrays of continuous-valued analog circuits for Łukasiewicz implication ($\supset$) and negated implication ($\not\supset$) circuits (Figure 2).



$$(\alpha \supset \beta) \supset (\gamma \supset \delta)$$

Figure 2. Fundamental ŁLA H-tree

Negated implication yields the simplest circuit. It is composed of a current source, a current sink, a three-wire summing junction, and a diode-connected MOSFET. The output is accurate to within 1% of the full scale and precise to within six to eight bits [9,10].

The dual logical and algebraic semantics of ŁLAS allows them to perform symbolic and numeric computations, a property essential to the applications described in this paper.

2: Sensors: the ŁLA retina

2.1: Retina design

A prototype ŁLA retina uses negated implication in a continuous-valued directed-edge sensor. This is a primitive vision function [11]. Four photocells are grouped in a cluster. Because negated implication is equivalent to a positive-only derivative, edges are detected in the NS, EW, SN, and WE directions across the cluster (Figure 3).

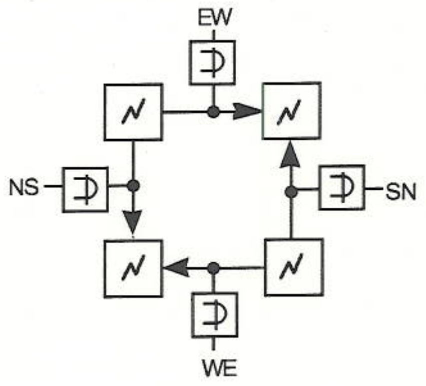


Figure 3. ŁLA retina four-pixel cluster

While the ŁLA retina could be implemented with npn and pnp phototransistors and diodes [12], our prototype uses npn phototransistors, MOSFET current mirrors, and diode-connected MOSFETs. It is still smaller and faster than Mead's retina [13] because the MOSFETs require less area than bipolar components (Figure 4).

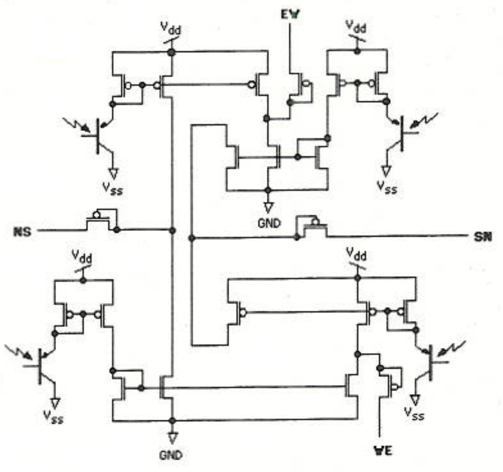


Figure 4. Four-pixel cluster schematic

2.2: Prototype retina EYE-1

An ŁLA retina prototype EYE-1 consisting of npn photocells and two edge detector strips was fabricated and tested to evaluate the design. Each photocell was laid out in a row consisting of other cells, each with the same size and shape. Two square photocells (bipolar transistors) were used, one 0.025mm high × 0.025mm wide, and the other 0.006mm high × 0.006mm wide (Figure 5).

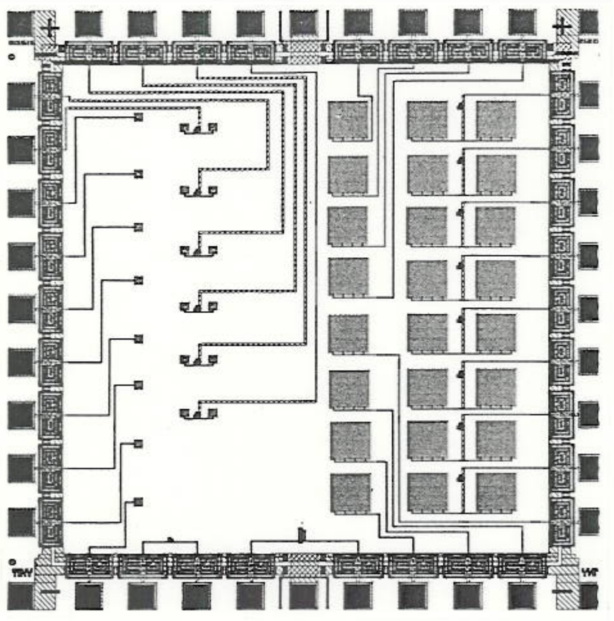


Figure 5. ŁLA retina prototype EYE-1

The edge-detecting capability of the prototype was characterized by projecting an edge onto one of the photocells, then measuring the edge voltages computed by a circuit whose inputs came from all photocells in the row. The selected photocell's response was thus measured relative to the output of the other cells in its row.

An 8mm movie projector lens focused light onto the exposed window of the chip. To project a sharp edge a strip of wire was attached to the front of the light source so that its shadow could be swept across the chip surface. A simple amplification circuit was constructed with a voltage output. A photocell was darkened by the edge, then the edge was adjusted to obtain the maximum output value possible from that cell. The output of all photocells was recorded, then the edge moved to the next cell.

The outputs were most distinct when the smallest cells were tested (Figure 6). This is probably because the edge was slightly wider than the width of the photocell. Since the large cells are much closer together there was probably some overlap which reduced the edge detection. The smaller cells have about 10 times more space between them and were less prone to this overlapping phenomenon.

The retina is also fault-tolerant. It will continue to operate if a MOSFET fails in one of the sensor clusters, or if several clusters fail entirely. The output of prototype retina EYE-1 indicated that variations in the photocells due to fabrication affected their sensitivity, but did not prevent the devices from functioning.

Measurement of the prototype retina EYE-1 indicates that its mimimum response time is limited by the response of the photocells rather than the negated implications. Photocells respond to edges in 20 ms, allowing all edges in an image to be identified in 20 ms. A 100-cluster retina

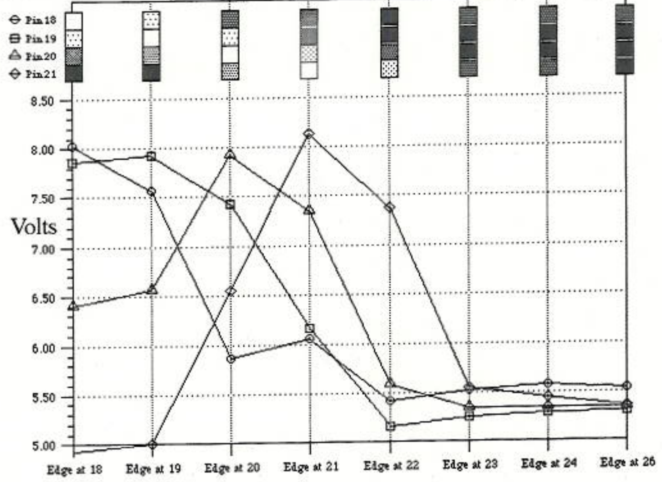


Figure 6. Output voltages from retina prototype

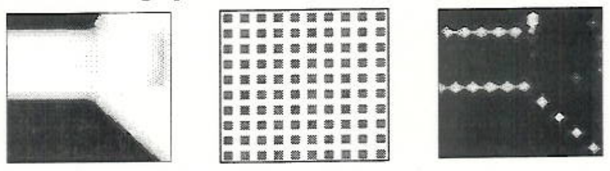
could detect up to 5000 edge-crossings per second in a dynamically changing image. This is four orders of magnitude fewer than MOSFET negated implications can handle [12], suggesting a direction for improvement.

The retina has some disadvantages. It is difficult to extract the results of the computation because the number of output pins is limited. If the data desired consists of a complete pixel map of edges, then each bit must be multiplexed to the output pins.

Another solution, used in the design of the controller, is to compact the results by merging computation with the sensor array. This type of device is known as a *computational sensor.* In this case, the edge data is compacted by computing the leg control signals, and outputting them instead.

2.3: Retina simulation

The ŁLA retina was simulated using a spreadsheet program. Sensor clusters are arrayed to form a 10×10 retina (Figure 7b). Each cluster was implemented with eight spreadsheet cells, four for each photocell in the cluster and four to compute the edges detected by negated implication. A graph of the input photocells reconstructed the image (Figure 7a), with a second graph used to visualize the edges (Figure 7c).



(a) Image (b) Retina (c) Output

Figure 7. Simulation of ŁLA retina

The difference in pitch between horizontal/vertical clusters and diagonal clusters results in a stronger response to horizontal and vertical edges (Figure 7c). The retina also generates noise by detecting weak edges (such as the patch on the wall in Figure 7a).

3: Control: a simple ŁLA subsumption architecture

A simple subsumption architecture for navigating a maze was implemented by breaking the necessary behavior into simple tasks. The tasks included walking in a straight line, turning left, turning right, and avoiding the maze wall. Even these few tasks are sufficient to generate simple emergent behavior, such as escaping from corners and backtracking. The controller for this subsumption architecture is implicit. It is implemented by partitioning the sensor clusters into regions that correspond to the tasks in the subsumption architecture (Figure 8).

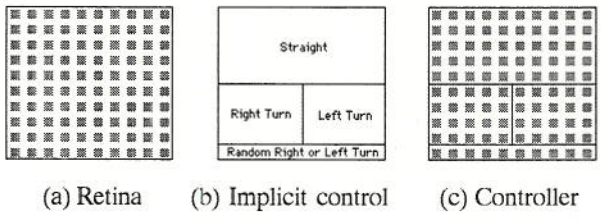


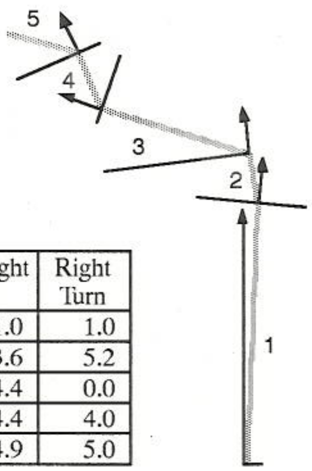
Figure 8. Subsumption architecture controller

Summing the edge signals in a given region results in an output signal for a specific task. The outputs are used to modify the robot's gait. Different tasks are activated as edges in the visual field move from region to region on the retina. A random left or right turn, a right turn, a left turn, and walking in a straight line compete for control of the robot based on the strength of each region's output. The random turn prevents the robot from getting stuck in corners and at walls. The robot's motion produces feedback that affects its behavior.

A simulation of Stiquito navigating a maze using a similar controller was developed for the Silicon Graphics Reality Engine [14]. Images from the simulation were digitized and transferred to the retina simulation. The sequence shows a robot moving through a maze (Figure 9a 1-5). As the image of the wall changes (Figure 9b 1-5) the edges move through regions in the controller (Figure 9c 1-5) to generate the tasks of the subsumption architecture (Table 1).

Table 1. Controller signals and path vectors

	Random Turn	Left Turn	Straight	Right Turn
1	0.0	0.0	21.0	1.0
2	0.0	5.0	3.6	5.2
3	10.0	0.5	4.4	0.0
4	2.0	1.0	4.4	4.0
5	1.0	4.0	4.9	5.0



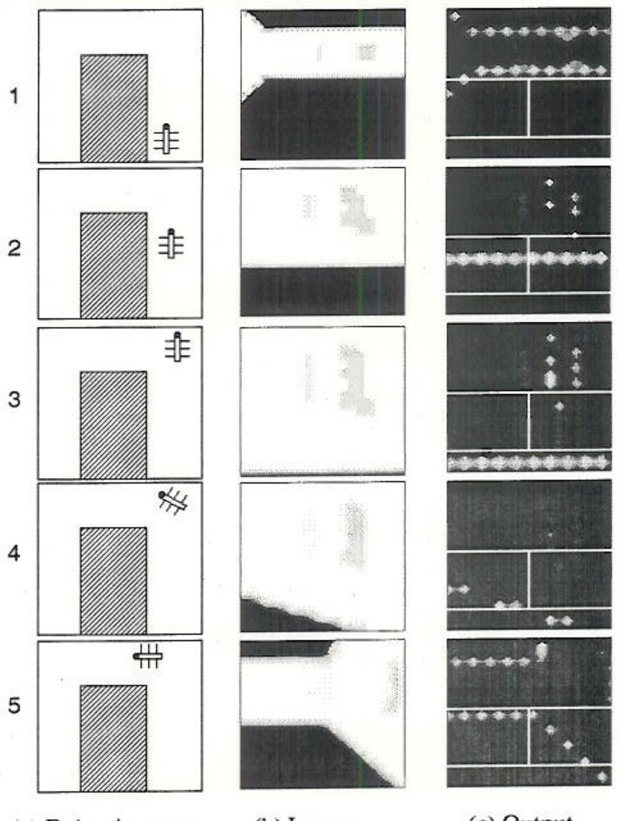


Figure 9. Maze navigation simulation

4: Locomotion: the ŁLA gait generator

4.1: Individual leg chaotic controller

Łukasiewicz logic can be used to specify chaotic systems [15]. By assigning truth values to components of a dynamical system Łukasiewicz logic can describe and control the behavior of the system. The ŁLA gait generator is constructed out of six identical chaotic controllers, one for each leg. The operation of a chaotic controller is defined by four functions controlling the trajectory of the foot: Backward(), Forward(), Raise(), and Lower(). The x- and y-motions are expressed as difference equations (Figure 10).

$$x_{t+1} = x_t + \Delta x_{t+1}\ ,\left\{ \Delta x_{t+1} \;\middle|\; \begin{array}{ll} x_t > y_t & \text{Backward}(x_t) \\ x_t \le y_t & \text{Forward}(x_t) \end{array} \right\}$$

$$y_{t+1} = y_t + \Delta y_{t+1}\ ,\left\{ \Delta y_{t+1} \;\middle|\; \begin{array}{ll} x_t > y_t & \text{Raise}(x_t) \\ x_t \le y_t & \text{Lower}(x_t) \end{array} \right\}$$

Figure 10. Self-referential sentences

The difference equations are mapped to Łukasiewicz logic arrays, and correspond to self-referential sentences of Łukasiewicz logic. Although the four functions may be interpreted as fuzzy functions, using fuzzy linguistic modifiers to define them did not work. Rule sets that combined functions such as *leg very far back* and *leg somewhat raised* ended up producing "stiff" leg trajectories with abrupt changes. When a smooth hand-drawn path was quantized, and the four resulting rules applied, a much improved trajectory resulted. The rules are shown graphically, scaled to the range -0.3 to 0.3 (Figure 11).

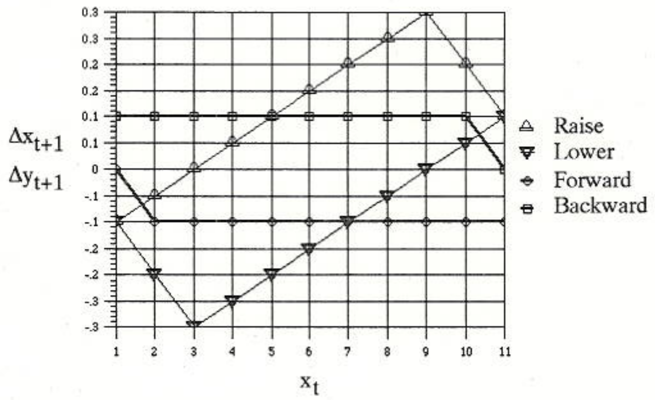


Figure 11. Leg motion rules

The rules do more than define a single trajectory. Because the rules define relative changes in the x and y axes, they apply throughout the space of truth values to define a state space containing a family of limit cycles. The limit cycles define multiple trajectories for the robot's foot in the state space (Figure 12).

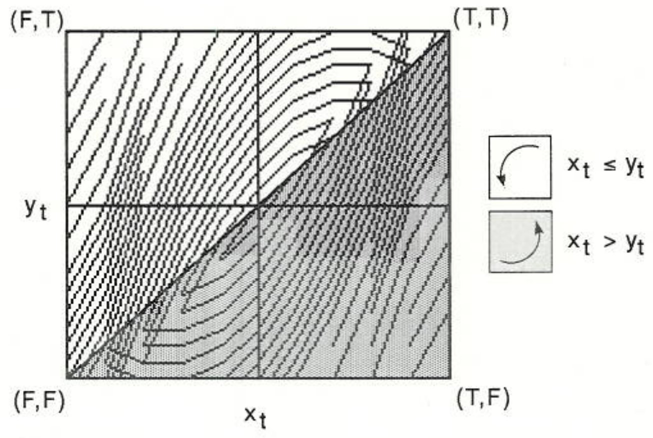
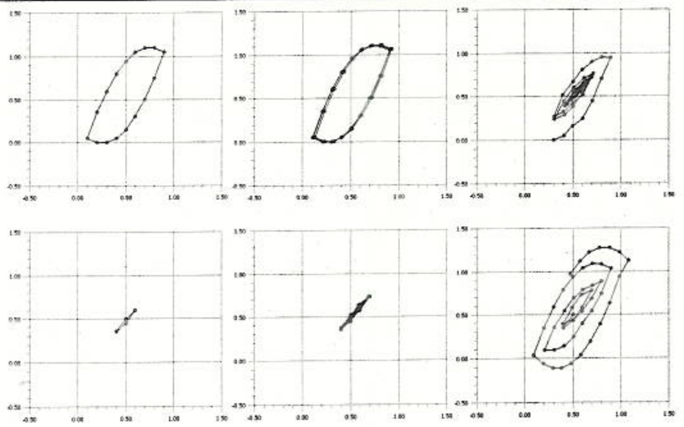


Figure 12. Limit cycles in the state space

A trajectory is stable if not perturbed (Figure 13a). Small perturbations, due for example to shot noise in the ŁLA circuit, will cause the trajectory to "jitter" (Figure 13b). Perturbations above a threshold $\approx \frac{1}{15}T$, such as a variable control signal applied as an input, will lead to increasingly chaotic behavior as trajectories alternate (Figure 13c).

Because the rules are evaluated algebraically, their effect extends outside the space of truth values if *min* and *max* are omitted from the valuation functions of logical connectives. This was done in spreadsheet simulations of the chaotic controller and the gait generator to view unclipped trajectories.

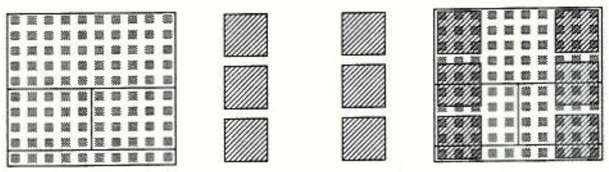


(a) Stable (b) η < threshold (c) η > threshold

Figure 13. Limit cycles perturbed by noise η

4.2: The gait generator

The gait generator for Stiquito is constructed from six chaotic controllers, one for each leg. The leg controllers are implemented with negated implication and packed into the unused space of the ŁLA retina. The computational sensor merges the ŁLA retina and the gait generator, and fits onto a MOSIS Tiny Chip (Figure 14).



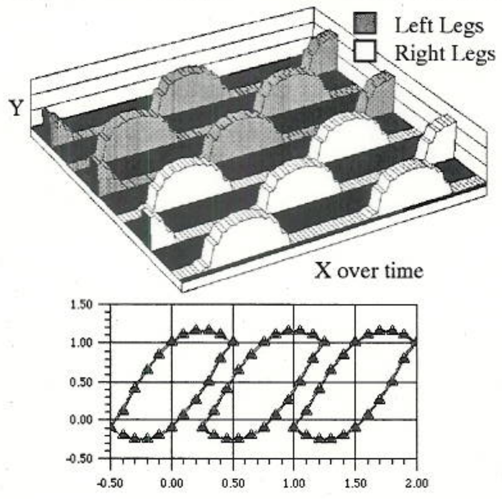
(a) Sensors & controller (b) Gait generators (c) Computational sensor

Figure 14. ŁLA subsumption architecture implemented as a computational sensor

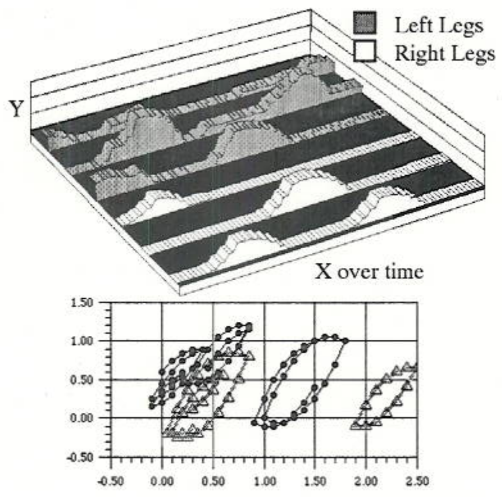
Because Stiquitos built to date have only one or two degrees of freedom per leg instead of four, two or three of the outputs of each chaotic controller can either be ignored or wire-summed to combine control in the x and y axes. Even so, the gait generator will produce a wide variety of gaits. Because the trajectories do not switch synchronously, out-of-phase oscillations in the unsynchronized trajectories produce many variations of the full tripod gait (Figure 15a).

Gaits that approximate single leg movement arise when a weak oscillation in five legs is coupled to a strong oscillation in a single leg. This is similar to the weak motion of the left legs shown in Figure 15b. Such a gait can be generated spontaneously by applying a strong perturbation to all legs until the trajectories are out-of-phase and degenerate. This is akin to many gait generators [16].

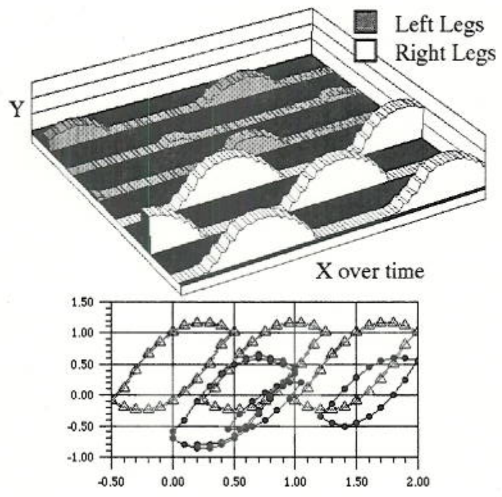
Turning gaits result when several legs on one side have strong oscillations, while legs on the opposite side have weak oscillations (Figure 15c). A turning gait is obtained immediately by forcing the legs on one side into out-of-phase and degenerate trajectories.



(a) Tripod Gait



(b) Noise-generated non-tripod gait



(c) Turning gait (quasi-tripod)

Figure 15. Gaits

5: Summary

Łukasiewicz logic unifies the design of sensors, control, and locomotion for a small, simple robot. The simplifications that result solve problems of weight, power, and complexity in the design of insect-like robots. The next step is to fabricate the computational sensor described here on a MOSIS Tiny Chip, and use it to control Stiquito.

Acknowledgements

This research was funded in part by NSF Grant MIP-9010878. Andrew Heininger fabricated the ŁLA retina prototype EYE-1; he and Amitava Biswas tested it. Shyam Pullela fabricated a programmable FPGA gait generator. Paulo Maciel developed a SGI graphic Stiquito simulator to study vision and machine learning in simple mazes. Jason Almeter suggested improvements to the implicit controller.

Access to this research

Instructions and parts lists to construct Stiquito; simulators for Stiquito, the ŁLA retina and gait controllers; and design files for FPGA controllers are available by anonymous ftp from cs.indiana.edu (129.79.254.191) in /pub/stiquito.

References

[1] Brooks, R. 1986. A robust layered control system for a mobile robot. *IEEE J. Robotics & Automation* **RA-2**(1).

[2] Brooks, R. 1990. A robot that walks: Emergent behaviors from a carefully evolved network. *Neural Computation* **1** (2): pp. 253-262.

[3] Beer, R. 1991. An Artificial Insect. *American Scientist* **79** (September-October): pp. 444-452.

[4] MacLennan, B. 1990. *Evolution of Communication in a Population of Simple Machines.* University of Tennessee Computer Science Department TR CS-90-99.

[5] Sanderson, I. 1965. *Living Treasure.* NY:Pyramid.

[6] Mills, J. W. 1992. *Stiquito: A small, simple, inexpensive legged robot.* Indiana University Computer Science Department TR 363a.

[7] Lukasiewicz, J., and A. Tarski. 1930. Untersuchungen über den Aussagenkalkül. *Comptes rendus des séances de la Société des sciences et des lettres des Varsovie Classe III* (23): pp. 30-50.

[8] Montante, R.A., and J. W. Mills. 1993. *Measuring Information Capacity in a VLSI Analog Logic Circuit.* Indiana University Computer Science Dept. TR 377.

[9] Mills, J. W., M. G. Beavers, and C. A. Daffinger. 1990. Lukasiewicz Logic Arrays. *Proceedings of 20th International Symposium on Multiple-Valued Logic*.

[10] Mills, J., and C. Daffinger. 1990. CMOS VLSI Lukasiewicz Logic Arrays. *Proceedings of Application Specific Array Processors*. Princeton, New Jersey.

[11] Ballard, D. H., and C. M. Brown. 1982. *Computer Vision*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

[12] Mills, J. W. 1992. Area-Efficient Implication Circuits for Very Dense Lukasiewicz Logic Arrays. *Proceedings of 22nd International Symposium on Multiple-Valued Logic*. Sendai, Japan: IEEE Press.

[13] Mead, C. 1989. *Analog VLSI and Neural Systems..* Reading, Massachusetts: Addison-Wesley.

[14] Maciel, P. 1993. Stiquito simulator for SGI.

[15] Stewart, I. 1993. Mathematical Recreations. *Scientific American.* February 1993: pp.110-112.

[16] Donner, M.D. 1987. *Real-Time Control of Walking.* Boston, Massachussetts: Birkhaüser.