

# Developing a Brain-like Computer with a Memory-Based Architecture

**H. Yamada, J. Takeuchi, T. Tominaga, T. Kato, N. Aibe, G. Matsumoto and M. Ichikawa**  
Lab. for Brain-operative Devices, Brain Science Institute, RIKEN, Wako 351-0198, Japan.  
<http://brainway.riken.go.jp/>

and

**H. Ono**  
Honda R&D Co. Ltd., 1-4-1 Chuo, Wako 351-0193, Japan

## ABSTRACT

We developed a simple model of an artificial creature equipped with a brain-like control unit. The model was a miniature car loaded with an FPGA (field programmable gate array) to implement its behavior decision-making circuits based on memory-based architecture. In this paper, we demonstrate the learning capabilities of a few of the systems implemented, and consider which of the learning systems had sufficient ability for the tasks at hand.

**Keywords:** Brain, Computer, Memory-based, FPGA

## INTRODUCTION

Recent research into the computing abilities of the human or animal brain is generally based on the construction of a brain-like computer and has attracted much attention [1-4]. Machines differ greatly from live creatures however, in that digital cameras must replace eyes, and silicon devices replace neurons. Currently, configuration of an artificial neural network similar to a human or animal brain is impossible, and the formation of the tremendous number of lookup tables required is not realistic. Our aim is the engineering realization of processing information in a manner similar to the brain, using conventional digital circuit techniques [1]. The indispensable capability of brain-like computing is the ability to make appropriate or quasi-appropriate decisions. This ability is based on experience, realized through the computer's interactions with the world, and is termed memory-based architecture.

The simple model for this preliminary study was to set a miniature car the task of smoothly navigating a course. We assumed real phenomena to be represented as coded symbols, and that learning is achieved through formulating linkages between input and output symbols. Trial and error is the main method of learning, until the final stage, when the linkages become fixed. Bearing in mind possible future research, we elected to design an actual miniature car with a real digital CCD camera rather than use simulation, in order to remain aware of practicalities such as redundancy, vibration, and so on.

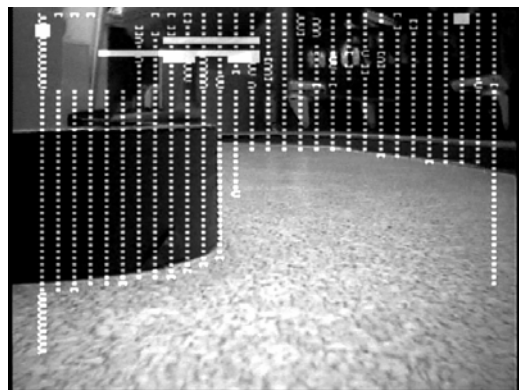
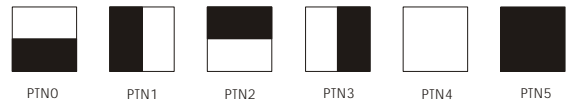
### 1. VIDEO PROCESSING and SYMBOL CREATION

We designed our miniature car (named IQ30) with a field programmable gate array (FPGA) and CCD cameras. The analog video input was converted to digital data through an analog to digital converter (ADC) and processed in the FPGA.

A logic programmer in the FPGA processed the input video array of 256 x 256 with 8-bit brightness or grayscale. Each pixel was averaged 16 times in the space domain. Several of the modules that are described below created symbols, representing some external physical parameters such as object distance, track curvature and so on. The logic programmer also considered the output to be directed to the steering servo or the DC motor as symbols. The initial operation of the car was to create connections between input and output symbols. The details of processing video from raw input to symbol creation are as follows.

### Primitive Pattern Matching

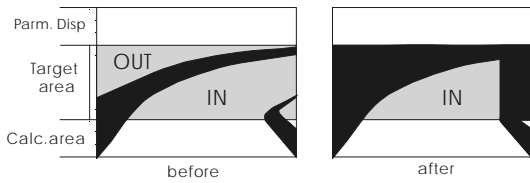
Pre-processing is accomplished by a primitive pattern-matching module, which converts the 256x256 pixel array to 64x64 primitive patterns, as shown in the figure below. There are six procedures operating in parallel for each primitive pattern in this module. The module also contains a sub-module that decides the proper threshold level, based on averaged frame brightness.



### Wall filter module

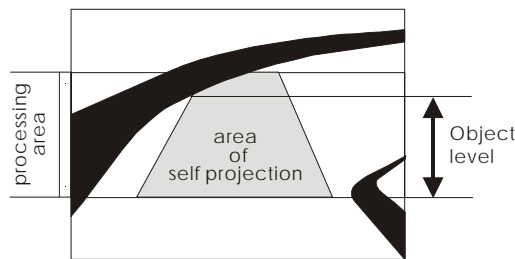
In pre-processing the video, the ability to distinguish between areas that are on or off the track (see figure) is also important. The FPGA memory stores 32x32 primitive patterns, and this module compares every possible vertical primitive for each location. Memory contents are switched to other primitives if the location is judged as being off the track.

Therefore, the following symbol creation modules use simplified primitive maps that contain only two categories: the wall or the track.



**Object Detection Module**

This module produces the symbols that are required to determine an object's distance in front of the car. A memory module contains primitives of the projection area in front of the car (see figure). A judgment logic unit counts the number of primitives representing the track, and determines its output according to this number. This number is used as the distance to the object. The output is a symbol for the object, which has sixteen degrees of resolution, ranging from 13-75 cm in front of the car.

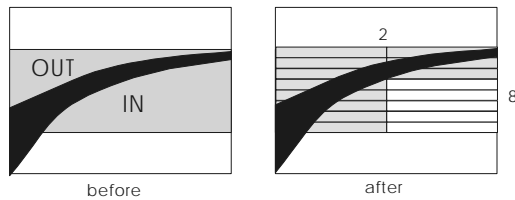


For distances closer than 13 cm, another resolution range (4 degrees for 0-13 cm) was adapted to detect the object.

**Road -Patterning Module**

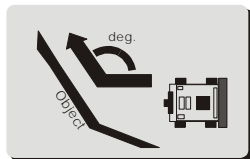
This module, and the two following, are related to the shape of the course and produce symbols for each module character.

The road-patterning module reduces the size of the memory contents from 32x 32 primitives to 2x8 black and white primitives (see figure). This new 16-bit pattern includes rough parameters for the distance to a wall from the car, the track's angle in relation to the car's direction, and so on.



**Trial Angle Detection Module**

Symbols from this module include rough information on the angle between the wall and the direction of travel. This module compares (exclusive OR logic) the right and left patterns of the 16-bit pattern from the road-



patterning module above. The output is symbols with 8 degrees of resolution.

**Direction Find Module**

This module simply counts the valid road area (in 32x32 primitives) and makes an output decision for the road ahead based on which area of the symbol (right- or left-hand side) contains larger numbers.

**Decision Making Module**

Here, connections are made between input symbols and output symbols (see figure). The basic procedure is as follows:

- i) The module makes an initial output to initiate an action. This initial output is a result of the initial linkage states between the inner input and output symbols.
- ii) The action is evaluated after it has taken place. This evaluation should be based on the programmer's overall purpose.
- iii) Linkages are changed, according to the evaluation.
- iv) The module creates an output representing the new linkages.

In this experiment, the evaluation was based on the signal sent from the velocity detection module when the speed was zero, as the programmer's main purpose was for the car to run for as long as possible.

An evaluation was made if the car stopped but was still on the track, which would cause the module to create a positive emergency symbol. This was usually when the car was stopped against a wall, after it had crashed into it. We programmed a sequential action for this case, whereby the car was to randomly move forwards and backwards with a random steering axis. This allowed the car to escape from the emergency state. Linkage changes were allowed to occur during emergency periods.

**2.THE OTHER INPUT**

**Velocity Detection Module**

The velocity sensor on the IQ30 was a combination IR LED and photo-diode, and it derived an inner symbol representing its velocity. The measurement range was about 0-3.0 m/sec. If the speed was zero (0 m/sec), the module produced an error signal.

**Color Detection Module**

This was an optional module for color input. It was not used in this experiment.

**G-sensor Module**

We also included a G-sensor; however it was not used in this experiment. The sensor is a two-axis G-sensor, designed for two-dimensional flat surface direction measurement. We plan to use this information to aid in the determination of other symbols.

**IR-sensor Module**

An infrared sensor was used to detect objects that were near to the IQ30 but not in front of it. These are complementary sensors to the CCD camera, which does not view to the side.

In this experiment, the infrared sensors were working, but they were not used.

### 3. EQUIPMENT DETAILS

**Dimensions:** L 200 mm x W 80 mm x H 70 mm

**Weight:** 370 g (battery included)

**Maximum speed:** 2.0 m/s

**Minimum turning radius:** 375 mm

**Battery:** 7.2V Ni-Cad DC battery (350 mAH)

**Maximum running time:** 15-20 min.

**Processor:** FPGA (xilinx XC400) 160 kGate used for this experiment (max. 400 kGate)

#### Sensors

CCD units (commercially available):

CCD: ¼ inch color CCD

Interface: NTSC

Frame-rate: 16.7 msec (1/60)

Resolution: H512 x V492 pixels

Power consumption: 130 mA

Lens: f4.6, field angle 40 degrees (horizontal)

Main clock: 19.06 MHz

Velocity sensor (custom made)

Measurement range: 0-3.0 m/s

Resolution: 256 degrees

G-sensor\*(included on the custom main board)

Chip: Analog Devices ADXL202

Axis: 2-axis (direction of travel and perpendicular)

Measurement range: -2G to +2G

IR-sensor\* (custom made)

Measurement range: 0-200 mm perpendicular to travel direction, i.e., sideways

#### Memory

Inside FPGA: dual port 20 blocks (512 byte/block)

Outside SRAM\*: Toshiba TC55V8200FT (2 Mword - 8 bit)

Outside DRAM\*: Toshiba TC59SM716 (2 Mword x 4 bank - 16 bit)

SmartMedia\*: Toshiba TH58V128DC etc.

**Steering Servo:** Futaba (S3101) 28x13x29.7 (mm) 0.18 sec/60 deg

**Motor:** Mabuchi FK130SZ

**Amp:** sanwa (ES01)

#### Interfaces

ISA (with custom board) or printer port for configuration of the FPGA (XC400)

IEEE1394 (in development): high-speed communication for global purposes

Optical LED

Speaker for sounds (representing inner states)

**ADC:** Analog Devices AD9280 (8 bit - 32 MSPS)

\*) Not used in this experiment

### 4. EXPERIMENT

Three experiments were made in order to determine linkages. The three experiments differed in the degree of randomness induced. In experiment A, the links were fitted in a totally random fashion. Experiment C was tightly scheduled to determine the links as set by a programmer. Experiment B was a combination of A and C. One important restriction was the trial time, which was limited by power consumption and battery capacity. There should be more emphasis on this kind

of physical restriction in learning-scheme discussions, and we return to this later. This kind of hardware simulation has inherent uncertainties or non-linearities that are difficult to replicate in software simulations.

The summary of our inner symbols is as follows.

#### Input symbols

a. Object: OBJ\_A (16 degree of 13-75cm), OBJ\_B (4degree for <13cm)

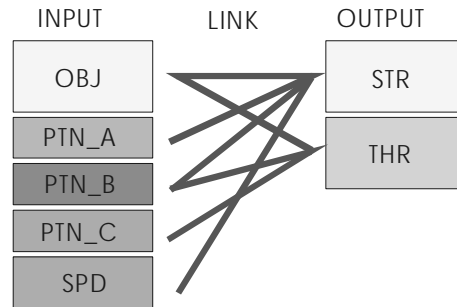
b. Shape of a Course: PTN\_A (a reduced road pattern: 16bit), PTN\_B (a trial angle the wall: 8 degree), PTN\_C (a direction of road: 1bit)

c. Speed: SPD (a velocity of the car: 32 degree)

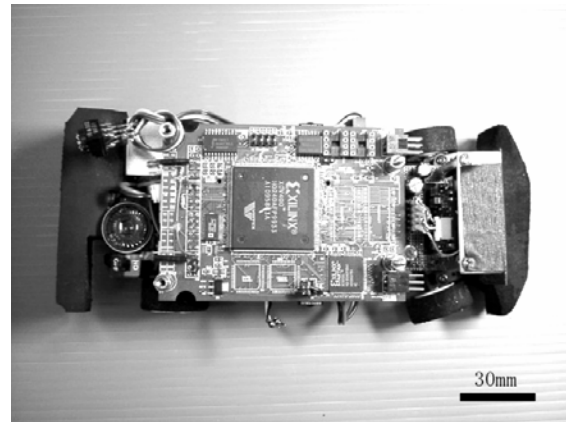
#### Output symbols

d. Steering: STR (8 degrees each for left and right)

e. Motor: THR (16 degrees, including backwards)

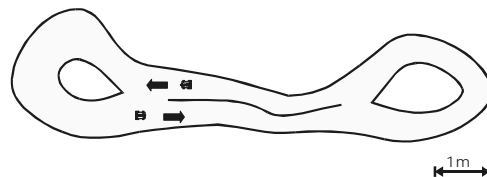


#### A photo of IQ30



#### A Course

Experiments were performed on a course in our laboratory (see figure). Total length of the course is about 20m and its width is from 40-100cm. IQ30 runs at 14sec for a lap on average after a complete learning. He can go on running for maximum 20min with 7.2V 350mAH battery.



**Experiment A (free-binding case)**

1. The links were connected in a totally random fashion. IQ30 then acted according to the links.
2. An error signal caused IQ30 to switch to emergency mode and the links were then modified randomly.
3. Trials lasted 10 min each. Ten trials using the above conditions were completed.

**Qualitative Results for experiment A**

Efficient links that would enable IQ30 to run smoothly were not created in any of the trials.

**Experiment B (slight-biding case)**

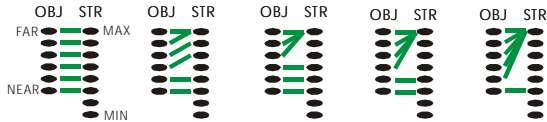
Similar to the conditions of experiment A, except that the speed was limited to about 1.4 m/sec. Also, a programmer set the link between PTN\_C and STR correctly. The other symbols were randomly linked during each trial.

**Qualitative Results for experiment B**

A few trials were successful, with the car running for over 2 min without touching any walls or objects.

**Experiment C (tight-binding case)**

In addition to the conditions in experiment B, a programmer prepared several typical links between OBJ\_A and STR (see figure). Each trial gradually fixed the links, and the programmer created a waiting list for modification of the links according to their importance. PTN\_B (wall trial angle) was used to determine the sensitivity of STR (steering angle). This determined the sensitivity of the steering when a low angle was detected between the wall and the direction of travel.



**Qualitative Results for experiment C**

More than 80% of the trials were successful, with the car running for over 2 min without touching any walls or objects.

**5. DISCUSSIONS**

The above scheme for symbol linking is fairly typical of the methods used in general learning problems. However, experiments performed with real hardware, like the IQ30,

experience several real physical constraints that limit learning time. In our experiment, a programmer aided the learning of linkages by adding meaning, and a degree of importance, to each symbol.

Based on this, we are planning the next IQ30 experiments with two main proposals.

**5-1. A random-link extraction method**

In our current experiment, a programmer had already prepared some symbol-producing modules. However, the next version of IQ30 will not require prepared modules. The programmer will only need to describe some empty modules that produce symbols; the IQ30 should then find the links from low-level symbols to high-level symbols independently. This means that the IQ30 will gradually create modules for higher symbols by linking low-level symbols. The linking structure will be expanded to a layered structure.

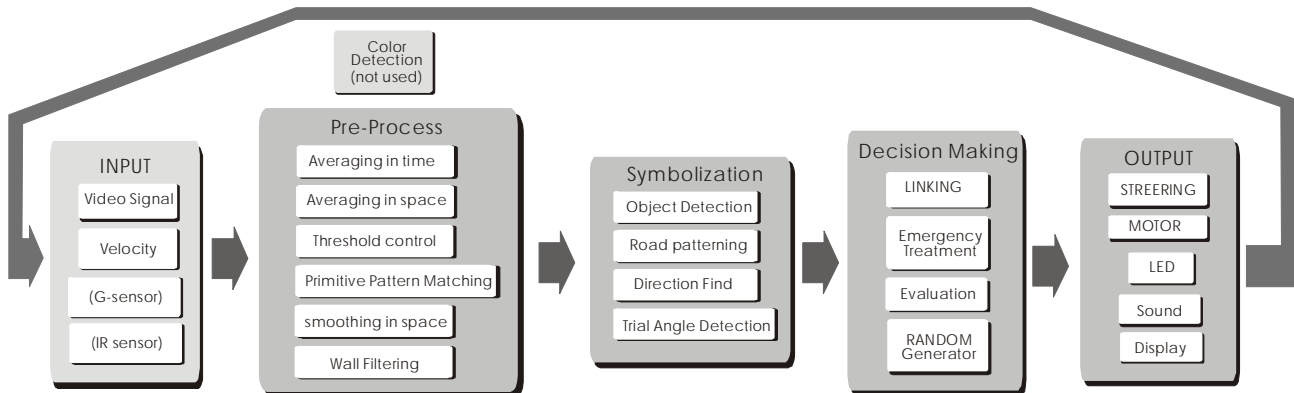
This scheme will be achieved by IQ30 repeating trials of the same task, creating symbols. IQ30 must make comparisons and find the similarities and differences between the low-level symbols in each trial.

**5-2. A detailed case-classification method**

In our current experiment, the only evaluation signal was one that indicated that the speed was zero. We believe that more efficient signals or structures would benefit this scheme, and propose that there should be some consistency between IQ30's tasks. In other words, previously determined decisions and the results of actions would be good evaluation factors for a learning problem. We suggest that IQ30's tasks should include ensuring correct positioning on a curve by ensuring that results are translated into symbols similar to the shape of the corner. A matching module, between the present symbol and the stored symbols for a particular corner, could create other symbols that are consistently related and are useful as evaluation signals. The next IQ30 should therefore store some specialized symbols classified according to their specialization. A programmer would then describe the initial structure for the classification. Detailed experiments from IQ30's trials would be stored automatically. This creates evaluation signals that are more suitable to evaluate learning.

There are obvious problems in these two methods. The random-link method will take appreciably longer to complete all of its linkages properly, while the classification method requires the programmer's knowledge for its initial

**Schematics inside FPGA**



classification. We are currently working to solve these problems through a hybridization of the above, or more exotic, methods.

## **6. REFERENCES**

- [1] M. Ichikawa, H. Yamada, G. Matsumoto "Realization model for brain computing", Applied Mathematics and Computation 111, 2000, pp.193-202
- [2] Y. Shigematsu, H. Okamoto, K. Ichikawa, G. Matsumoto "Temporal Event Association and Output-Dependent Learning: A Proposed of Neural Molecular Connections", Journal of Advanced Computational Intelligence, Vol.3 No.4 1999
- [3] G. Matsumoto, "The Brain and Brainway Computer", Proc. 5<sup>th</sup>. Int. Conf. On Soft Computing and Information/Intelligent Systems 1:13-20, World Scientific.
- [4] E. Koerner, G. Matsumoto "Cortical Architecture and Self-referential Control: How the Brain Organizes Computation", IEEE BME (in press)

