Proceedings of the 1991 IEEE
International Conference on Robotics and Automation
Sacramento, California - April 1991

VLSI Sensori-motor Systems

James J. Clark
Daniel J. Friedman

Division of Applied Sciences
Harvard University
Cambridge, MA

Abstract

It is our belief that the bulk of the sensors intended for use in robotic applications should be designed taking into account the requirements of the robots motor systems. To this end one should design special purpose sensing arrays which output motor signals rather than sensor signals. Sensors of this kind would greatly reduce the computational burden from that imposed with standard sensing techniques that use high bandwidth video cameras or even tactile sensing arrays. In this paper we describe our efforts to date in developing sensori-motor chips which contain arrays of sensing elements, circuitry for processing the raw sensor data into forms relevant for motion related tasks, circuitry for generating motion signals based on the processed sensor data and the goals of the system, and finally which contain an operating system which selects a unique motor command from a set of usually conflicting motion signals.

1 Introduction

In most robotic systems sensory information is used to generate manipulator motions. Currently this involves (in the case of visual sensing) imaging the robots workspace with one or more video cameras, digitizing the high bandwidth video signals from the camera(s), performing image processing and analysis operations of varying complexity on the digitized data, and then reasoning about the processed data to determine a suitable motor activity.

The amount of computation required in the above scheme is exceedingly high. One reason for this is the general purpose nature of the sensing device. The video camera has a high bandwidth output as it contains a large amount of redundant information, as well as information that is irrelevant to the task at hand. Similarly the computational components of these robotic systems are often general purpose, and hence inefficient for doing a particular task.

One way to reduce the amount of computing machinery needed in a robotic system is to reduce the bandwidth of the sensory device. This can be done by integrating in with the sensor array circuitry for extracting only relevant and non-redundant portions of the sense data. In this paper we describe sensors that we have built which combine sensing arrays with spatial filters and simple feature extractors. We have also developed an approach to the design of intelligent low bandwidth sensors based on an active sensing paradigm, which will not be described here. Details on this rather unconventional approach can be found in [5].

We can do even better than this, however. In robotic systems one does not really care about the sensory information itself; one is more concerned with the actions that the robot is to perform. Each robotic system is performing sensing to help carry out motions, whether these motions be navigational (exploration, transport, etc) or manipulatory (handling objects, assembly, etc). For each of these motor tasks there is a corresponding set of sensory tasks that need to be done. Besides purely manipulatory tasks, sensing tasks of various kinds also have motion inherent to them. For example, active vision and tactile sensing algorithms require sensor motions in order to perform properly. The sensing devices, to be most efficient, should output *motor* information, and not *sensory* information. Thus we have been investigating the implementation of *sensori-motor* devices in VLSI technology. These devices contain arrays of sensors (which are typically less dense than a video camera sensor array) and circuitry which processes and examines the raw sense data and outputs a low bandwidth stream of motor signals. Many of these chips can be integrated in a sensori-motor system which handles the multiplicity of different motor requests coming from the sensorimotor chips depending on the goals and current state of the robot.

Even on a single chip there may be multiple "sensor tasks" which output differing, and usually conflicting,

motion requests. These motion request must be handled by an "operating system" of some kind to produce a unique motion command. We describe a simple, extensible, operating system, similar in flavor to Brooks' subsumption architecture [2], which can be embedded on the same chip as the sensors and the sensor processing circuitry.

2 Integrating Sensors and Sensor Processing Circuitry

The first stage in any sensori-motor system is processing the raw sensor data so that the information relevant to the motor task is extracted from the sensor data. There has been a great deal of interest in combining such sensory information processing circuitry and sensors in a single integrated circuit. A currently popular technique is that of using a resistive grid to implement forms of temporal and spatial filter [6, 8]. This approach has the advantage of being relatively straightforward to implement and many image analysis operations map naturally into this paradigm. It has a number of drawbacks, however, one is that it is limited in the types of operations it can implement; another is that the implementations of some operations are unstable [9]. In this section we describe a rather straightforward approach that we have been investigating. This approach uses current-mode circuit techniques (see for example [7]) to implement spatial convolutions directly.

The spatial processing of current mode signals is based on Kirchoff's Current Law, which states that the sum of the currents flowing into and out of a circuit node must be zero. This law allows us to perform spatial convolution in a very straightforward manner by summing weighted currents from spatially disparate sensing elements at a summation node. If the sum of these currents from the sensors does not add up to zero we will need to add an additional current from somewhere to make this sum go to zero. This additional current will thus be equal to the negative of the sum of the inputs currents. We can supply this current from a coupled pair of n channel and p-channel current mirrors which will in addition, provide differential, weighted copies of this sum which can be used in further processing. A schematic of this current mode convolution scheme is shown in figure 1. The details of the operation of the convolution element can be found in [4].

The convolution cell provides a voltage that is proportional to the sum of the input currents provided to the cell. Since these inputs are scaled replicas of currents produced by sensing elements, the output voltage

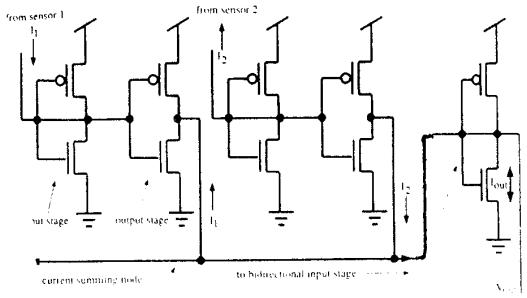

Figure 1: The current mode convolution element.

of the convolution cell has performed one point in a spatial convolution on the output of the sensor array. One could be satisfied with the output voltage obtained from the convolution cell and output it through a buffer amplifier and pass it off chip. This would then allow one to do single pass convolutions on the outputs of the sensor array. However, the nature of our convolution elements allows one to perform multiple cascaded convolutions. For example, one could implement oriented bar detectors.

We have built a number of sensor arrays which contain these current mode convolution circuits using a 2 micron double metal, single poly, CMOS process through the MOSIS foundry. A cifplot of one of these chips is shown in figure 2. This particular chip (2.25mm by 2.25mm in size) contains a 5 by 5 array of phototransistors as well as circuitry for computing three different convolutions (in parallel). These convolutions include an x-derivative, a y-derivative and a Laplacian operation. The convolution is done only for the central point. The chip contains a form of automatic gain control which reduces the effect of common mode (with respect to the sensor stimuli) variations. This circuitry, which is based on using an average of the sensor signals to produce an adaptive bias signal, allows responses of the convolutions over a larger range in light levels. There is also circuitry on chip for processing the image data to produce motor signals, and this will be described in a later section. We have also fabricated and tested arrays which contain different types of convolution kernels, both 1D and 2D, computing x and y derivatives and smoothing of images. These chips have been tested and shown to implement the desired operations, proving the effectiveness of the current-mode approach. The current mode technique is not limited to optical sensors. We have also fabricated a 7 by 6 element magnetic sensor array with combined smoothing and Laplacian convolution [4]. This chip was intended

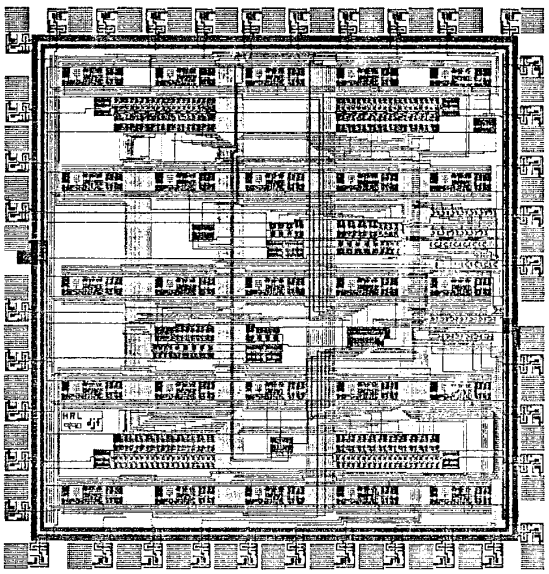

Figure 2: Plot of a 5 by 5 element visual sensori-motor chip

for use in a magnetic field based tactile sensor [3].

3 Monolithic Sensori-motor Systems

As mentioned in the introduction, most of the sensory information provided by a high resolution video camera is irrelevant for many robotic tasks. What a robotic sensor should provide is a signal that is substantially reduced in complexity, to ease the computational burden on the system that is acting on the sensor data. In addition, in many robotic situations, such as in the manipulation of objects or in workspace exploration, the system that takes in sensory data ultimately produces a motor signal of some kind. To this end our system needs to do two things. First it must convert the processed sensory data into a motion signal. A robotics system may contain many of these sensing-to-motion operations in parallel, each trying to satisfy a particular goal of the robot. Thus, the second thing that our sensori-motor system must do is choose between these motion signals and produce a unique motor signal that can be sent off to the drivers of the robot actuators.

We will now present an architecture, suitable for integration with sensing elements for implementing such a sensori-motor system. This architecture is based not on Brooks' subsumption ideas, but instead relies on an interrupt-driven, active sensing-based design.

The architecture consists of a number of distinct components. The first stage is the raw sensor array. The output of these sensing elements is processed by a set of modules which extract information relevant to the sensory task modules which follow. The sensory task is a subsystem which attempts to get the overall system to meet a particular small set of objectives, or indeed a single objective. Sensory tasks include motor components. Specifically, a sensory task continually suggests courses of action for motor systems which the task module itself calculates to be most appropriate for the completion of its small, built-in mission. It is always active in that it is always processing data and suggesting actions. The courses of action suggested by a sensory task take the form of motor signals, which, in their complete form, come in sets. In each set of such signals there is a data channel dedicated to each and every motor in the system.

One example of a sensory task could be edge location. This task might be accomplished using very simple behavior, such as suggesting motion straight ahead for the overall system until the edge detector sensory primitive is activated. A second example of a sensory task could be centering the overall system over an edge. In order to achieve this task, we might first postulate a sensory primitive which locates edges on the sensor array, and another which reports whether an edge is centered on the array. The sensory task could then be implemented simply in terms of the two primitives, namely, move in the direction of the edge until the edge is centered, then stop. In order to make sure that the system does not try to implement this procedure when no edge is present, the task could be designed so that it will return an error signal unless an edge is detected in the sensor field by the edge detection primitive. Sensory tasks clearly rely heavily on sensory primitive data for their function.

Sensory tasks are most useful as building blocks for more complex procedures. Sensory tasks take simple information which is easy to acquire from sensors or sensory primitives and generate output which is "atomic" in nature. By atomic, we mean that the output from sensory tasks applies to very basic situations and relates to the achievement of very small, finite goals. The atoms, namely, the sensory tasks, serve as building blocks for high-level overall system behaviors. Ideally, sensory tasks are designed to be sufficiently flexible so that they can be used in multiple contexts to help achieve ranges of more complex objectives. One sensory task which is easy to see in this light is the task of edge location. This task is important in virtually every object location, landmark

identification, and navigation routine imaginable where standard vision is the sensory modality. Edge location sensory tasks, then, would be used over and over by higher level modules.

The sensory task modules invoke motor tasks to help get their job done. The motor task is an architectural element which is closely related to the sensory task. Instead of focusing on the detection of particular phenomena in the environment, however, the atomic elements in a motor task involve the implementation of a set of motor commands as a unit to accomplish some simple goal. Again, the motor signals which comprise the output of a motor task come in sets, with each motor receiving data through dedicated channels. One motor task might be as simple as moving straight ahead. Another might be circular motion. These tasks could be either locked into place, only escapable through some as-yet-undefined interrupt procedure, or could be ended after a predetermined interval of time had elapsed, or could be terminated through a change in motor state recognized by the use of proprioceptive senses. The idea behind the motor task concept, like the idea behind the sensory task concept, is that the task be simple and usable as a building block for various types of more complex behavior. The motor task is distinct from the sensory task in that it does not sample environmental data in order to make its decisions. It is quite parallel to the sensory task, however, in that the concept underlying motor tasks and sensory tasks is identical, the only difference being that the environmental data used by the sensory task is replaced in the motor task by proprioceptive data.

We have described sensory tasks and motor tasks as atomic elements of the overall architecture. Goals are defined as the structures which serve to combine sets of those atomic elements into larger routines. These routines, then, are designed to accomplish some objective which is more complex than one met by a sensory or motor task alone. Goals form the layer in the overall system at which overall system behavior is truly specified in the sense that the set of goals is the set of macroscopic objectives the system will be able to perform, given appropriate circumstances.

An example of a goal which a simple autonomous system might implement is "follow an edge." This goal can be subdivided into a number of atomic sub-goals and thus can be described in terms of sensory and/or motor tasks. The first step in following an edge is finding an edge. One way that finding an edge can be implemented is as a sensory task which sends motor requests for straight ahead motion until the edge-detection primitive is activated, then sends a motor request for stopping and finally signals its completion to higher processing levels. The next sensory task which would be activated in edge-following would be one which orients the system along an edge. This task would calculate and send motor signals designed to center the system over the edge. The centering would be carried out in such a way so that future forward motion will carry the system parallel to the edge as currently observed once centering is complete. The edge-following objective could then be continued by alternating between a simple motor task, move forward either a fixed distance (using a proprioceptive odometer-type sense) or for a fixed time (using an internal timer), and a sensory task, the centering task described above. The motion task sends the system along the edge and the centering task recenters the system over the edge. The goal would remain active until the system lost track of the edge, as indicated by data from the edge detection primitive, or until an interrupt signal preempted the goal.

So far, we have seen that goal modules are connected to the system primarily through motor and sensory tasks. Motor and sensory tasks suggest motor signals which will, if properly designed, help the system accomplish the atomic task objectives. The goal module has to arbitrate among all of these suggested courses of action. It does this in that only one of the task nodes which feed the goal module data are active with respect to that goal at any one time. The active task, of course, reflects the current state of the goal module. Knowledge of the preset order of tasks for a given goal as well as which task is active, then, indicates which among the atomic subgoals have been and which have not yet been accomplished. That one task is active with respect to the goal implies that motor signals are received by that goal only from the active task. The other tasks are by no means inactive. In fact, as mentioned earlier, all tasks in the system are continually gathering data from sensors, sensory primitives, and proprioceptors. Furthermore, all tasks are continually calculating desired courses of action and trying to send those suggestions to all goal modules connected to them. Each goal in the system selects exactly one of its component tasks and accepts this task's motor signal suggestions as its own motor signal suggestion to be passed on to higher levels in the architecture. Note that it is entirely possible for a goal to receive input not only from task modules, but from other goal modules. Because the output from a goal, like the output of a task, is a stream or set of streams of motor signals, goals can receive input from other goals or from tasks equally well.

The final component of our architecture is the operating system (more details on our approach to this can

be found in [1]). Much as the goal module serves to arbitrate among task modules which compete for control of the overall system, the operating system serves to arbitrate among competing goal modules. Each goal module presents a course of action for the overall system in the form of commands sent to the system's motor resources. The courses of action suggested by each goal, of course, are generated at the task level as described earlier and are passed from task to goal, either directly or through further, higher level goals, and ultimately toward the operating system. The operating system, depending on its internal state, chooses a course of action by choosing which goal is active with respect to the operating system at any given time. Again, this does not imply that the goals which are not active with respect to the operating system are inactive. The active goal is the goal whose motor suggestions are being listened to and acted upon by the system at large; the operating system ignores the suggestions of the goals which are not active.

At its simplest level, then, the control system for the entire autonomous robot can be understood in terms of motor signal flow through binary gates from the task level to the operating system level. Each task sends a flow of motor signals upward to as many goal modules as are connected to it. The goal modules control a set of information passage gates, one for each task connected to a given goal module. Some goals may be connected to other goals as well; each goal feeding signals to a higher level goal requires that higher level goal to accommodate it with one additional information passage gate. One and only one of these information passage gates is open at any one time, insuring that each goal suggests one uncluttered course of action to higher levels in the architecture. Finally, some subset of the complete set of goal modules sends data upward to the operating system. The operating system is in many ways analogous to an overall top-level goal module. The operating system has one information passage gate for each goal module connected to it and passes exactly one set of motor signals to the motor-driver circuitry. Because the operating system listens to only one goal module at a time, it can only try to implement a single motor command set at a time, thus preventing the system from trying to do two mutually exclusive things at once. If necessary, the motor signal set could be processed in the operating system as well so that motor signals passed around by the lower system levels could modified, if necessary, to meet motor driver signal requirements.

As an illustration of our approach consider the system which we are in the process of constructing. The chip depicted in figure 2 contains part of a full sensori-motor system for an insect-like system. Its overall goal (which is similar to the "buggy" system described in our paper [1]) is to find a source of "food", which we assume to be found in cracks in the floor. The system will have the sensori-motor chip looking down at the floor, and will contain a motor system that will allow it to move about the floor with 3 degrees of freedom (x, y and orientation). It is further assumed that the cracks in the "floor" that the system moves about on will have sufficiently different light intensity levels as to appear as (pairs of) edges in an image of the floor.

The chip that we have built contains, as mentioned earlier, a 5 by 5 array of phototransistors, along with current-mode convolution circuitry for computing x and y derivatives and the Laplacian. In addition to this the chip contains logic circuitry that determines eleven quantities related to the sensor data:

- i) Whether an edge is present or not within the 5 by 5 sensor receptive field of the chip (1 signal).
- ii) If there is an edge, it determines the orientation of the edge, as one of eight possible values (e.g. 45 degree increments) (8 signals).
- iii) If the edge is determined to be a horizontal edge or a vertical edge the chip decides whether the edge is centered in the 5 by 5 sensor receptive field (2 signals).

These processed sensor signals are produced as follows. The analog outputs from the three convolution circuits (V for the x derivative, H for the y derivative, and L for the Laplacian) are passed to pairs of comparators, generating six digital signals. One comparator in each pair detects whether the signal is less than a low threshold (resulting in signals we call V^-,H^-, and L^-), the other comparator in each pair detects whether the signal is greater than a high threshold (resulting in signals we call V^+,H^+, and L^+). These six signals are then input to Boolean logic which computes the desired eleven signals described above. The logic equations for this process are given below.

$$0\_edge = \bar{E}(V^+L^- + V^-L^+)$$

$$45\_edge = (V^+H^-L^- + V^-H^+L^+)$$

$$90\_edge = \bar{E}(H^+L^+ + H^-L^-)$$

$$135\_edge = (V^+H^+L^+ + V^-H^-L^-)$$

$$180\_edge = \bar{E}(V^+L^+ + V^-L^-)$$

$$225\_edge = (V^+H^-L^+ + V^-H^+L^-)$$

$$270\_edge = \bar{E}(H^+L^- + H^-L^+)$$

$$315\_edge = (V^+H^+L^- + V^-H^-L^+)$$

$$vert\_centered = \bar{F}(V^+ + V^-)$$

$$horiz\_centered = \bar{F}(H^+ + H^-)$$

$$edge\_detected = F + vert\_centered + horiz\_centered$$

where

$$E = 45\_edge + 135\_edge + 225\_edge + 315\_edge$$

$$F = E + 0\_edge + 90\_edge + 180\_edge + 270\_edge$$

The signals produced by the chip can be used to produce the motor requests needed for attaining the goals of the system. For example, if the goal of our system is to find "food" and it is assumed that food can usually be found in cracks in the floor, then the following behaviour may be used.

- 1. Look for a crack in the floor. This is done by moving the robot in some specified search pattern if the "edge_detected" signal from the sensorimotor chip is low.
- 2. Once an crack has been found it will typically not be oriented such that motion of the robot in the same direction it was previously moving will follow the crack. Thus the robot must orient itself with the edge. The information needed to command the orientating movements come from the "edge_orientation" signal of the sensorimotor chip.
- 3. Once the robot has been properly oriented (e.g. so that the edge appears vertical to the robot), the robot must center the edge so that it may be reliably followed.
- 4. Once the robot has lined itself up with the crack it moves forward until the edge is no longer centered, or when the orientation is no longer vertical or when "food" has been found.

From the above sequence of steps it can be seen that the chip that we have built, simple as it is, still contains most of the circuitry required for implementing an autonomous system capable of non-trivial behaviour. The next chip that we intend to build will contain the operating system component which will take in motor requests from multiple such sensori-motor chips, each concerned with a different behavioural aspect, and output a unique motor signal to the robot actuators.

4 Acknowledgements

The authors gratefully acknowledge the support from the Joint Services Electronics Program, grant number N00014-89-J-1023 and from the National Science Foundation, grant numbers CDR-85-00108 (University of Maryland/Harvard University Systems Research Center) and NSF-IRI-90-03306.

References

[1] Bestavros, A.A., Clark, J.J., and Ferrier, N.J., "Management of Sensori-Motor Activity in Mobile Robots" Proceedings of the 1990 IEEE Robotics and Automation Conference, Cinncinati, May 1990, pp 592-597

[2] Brooks R., and Connell, J., "Asynchronous Distributed Control System for a Mobile Robot", *SPIE Proceedings*, Vol. 727, October 1986.

[3] Clark, J.J., "A Magnetic Field Based Compliance Matching Sensor for High Resolution, High Compliance Tactile Sensing,", Proceedings of the 1988 IEEE Conference on Robotics and Automation., Philadelphia

[4] Clark, J.J. and Friedman, D.J., "Local Processing of Sensor Array Data Using Current Mode Circuitry", Proceedings of the 1989 International Electron Devices Meeting, Washington D.C., December 1989, pp 507-510

[5] Clark, J.J. and Hewes, R.P., "Active sensing at a microscopic scale", IEEE Symposium on Intelligent Control, September, 1990, Philadelphia

[6] Harris, J., Koch, C., Luo, J., and Wyatt, J., "Resistive fuses: Analog hardware for detecting discontinuities in early vision", in **Analog VLSI Implementation of Neural Systems,**, C. Mead and M. Ismail (eds.), Kluwer Academic Publishers, 1989, pp 27-56

[7] Kawahito, S., Kameyama, M., Higuchi, T., and Yamada, H., "A 32x32 bit Multiplier Using Multiple-Valued MOS Current Mode Circuits," IEEE Journal of Solid-State Circuits, Vol. 23, No. 1, pp 124-132, 1988

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

[9] Standley, D.L., and Wyatt, J.L. Jr., "Stability criterion for lateral inhibition and related networks that is robust in the presence of integrated circuit parasitics", IEEE Transactions on Circuits and Systems, Vol. 36, May 1989, pp 675-681