ASP: An Algorithm and Sensor Performance Evaluation System[1]

Thomas C. Henderson, Chuck Hansen and Bir Bhanu

Department of Computer Science
The University of Utah
Salt Lake City, Utah 84112

Abstract

We describe a methodology which permits (1) the precise characterization of sensors, (2) the specification of algorithms which transform the sensor data, and (3) the quantitative analysis of combinations of algorithms and sensors. Such analysis makes it possible to determine appropriate sensor/algorithm combinations subject to a wide range of criteria including: performance, computational complexity (both space and time), possibility for concurrency, modularization, and the use of multi-sensor systems for greater fault tolerance and reliability. Some examples from the domain of remote sensing are given.

[1]This work was supported in part by NSF Grants ECS-8307483 and MCS-82-21750. Chuck Hansen is an ARO Fellow.

ASP: An Algorithm and Sensor Performance Evaluation System

1. Introduction

Current systems for the analysis of remotely sensed data are limited in their capability by extreme data dependency and a set of *ad hoc* algorithms [1, 2]. The performance of such systems in realistic scenarios is not very good and results in poor recognition accuracy and a high false alarm rate. One approach to solving this problem is to develop expert systems for the automatic analysis of imagery. For example, Nagao and Matsuyama [7], describe an expert system for the analysis of aerial photographs, and Tsotsos [8] presents an expert system for understanding motion in images. Although such systems may achieve a certain amount of success, we believe that, in order to produce better systems, it is essential to consider the system as a whole, and, if possible, to produce the system subject to known design constraints and requirements. All this requires a thorough understanding of the algorithms for low level data analysis such as preprocessing, detection, segmentation and feature computation. To a great extent the success of such an analysis system depends on the low level image analysis.

Recent efforts have been concentrated on applying a given algorithm to a set of images and carrying out statistical analysis without regard to the sensor. This is one of the root causes for the lack of understanding of the behavior of algorithms and the inability to predict their performance in real situations. In this paper we describe a novel framework which allows the evaluation of the performance of a suite of algorithms based on the interaction of algorithms with the sensor.

In a multi-sensor environment it is necessary to exactly characterize the nature of each sensor, including the type of information returned by the sensor and the manner in which this information is obtained [5, 4]. For example, most sensors can be characterized to some extent in terms of a small set of features such as error, accuracy, repeatability, drift, resolution, hysteresis, threshold and range. Moreover, as we are dealing with digital signal processing, measures of quantization and sampling performed by the sensor are also considered. In general the essence of any sensor is the domain over which the sensor operates and the kind of data it returns; e.g., a camera provides x and y spatial values (perhaps implicitly according to location in the signal) and light intensity information. A camera can be viewed as a function over its two-dimensional viewing space, or as a stream of triples (x,y,intensity) produced by it. A computational theory of sensors is explicitly based on the sensor, the algorithm, and the manner in which the algorithm acts on the output of the sensor. It requires the definition of the domain over which a sensor operates, the physical nature of the transduction, and the characteristic output of the sensor. Given this framework, it is possible to make a quantitative performance of various digital signal processing algorithms on the sensed data.

There are several difficult issues involved in choosing a scheme whereby features of algorithms can be composed with features of physical sensors such that the overall sensor system may be analyzed. A desirable characteristic of a complex suite of algorithms as required for analyzing remotely sensed data is that each of its component algorithms makes maximum use of the input data characteristics and its goals are in conformity with the end result of obtaining the best recognition performance. One approach to this problem is to view algorithms and sensors in much the same way; i.e., an algorithm takes in a certain set of data from a sensor or from another algorithm and produces a set of transformed data. Its output can then be characterized in much the same way as the output of an actual sensor. This view permits an economy of representation and easy movement from actual physical sensor devices to sensor/algorithm combinations. Not only does this make the analysis easier, but it also gives a mechanism by which it is possible to consider different sensor/algorithm combinations to be equivalent according to the characteristics of their output. As an example, if an edge detection algorithm is run on a certain kind of image data, then the output is essentially that of a "smart" sensor which detects edges directly. Similarly, two cameras and a stereo algorithm which fuses their data to produce range data may be considered equivalent to an actual direct range finding device. This also makes it possible to consider under what circumstances the sensor algorithm combinations are different and to use those differences to solve user or problem imposed constraints. For example, given two systems which ultimately produce data with the same characteristics, one system might be faster than the other or one might have better accuracy than the other.

Finally, it is necessary to provide a quantitative analysis of the performance of different sensor systems. This can be parameterized along whatever dimensions are defined for the various sensors and algorithms. A major difficulty in resolving such issues is presented by the great variety of sensor systems and the varying level of awareness of such issues within different sensor user communities. Experienced users of certain types of sensors may have a fairly good knowledge of when and why certain algorithms work well. However, algorithm evaluation techniques are not standardized, and there are many ways in which the properties of algorithms can be characterized. This is one of the major motivations for establishing a uniform framework for the description of both sensors and algorithms; namely, it becomes possible to define a coherent computational theory of sensors. Then, the sensor system can be defined not simply as a task to be done, but rather as a task to be performed in an optimal way. Of course, the optimal solution depends on the application.

The use of statistical, heuristic and parametric models is considered for algorithm evaluation on a sample database of a given scenario. In this context these models are chosen such that each part of the remote sensing system can be evaluated not only with respect to its own figure of merit, but also against the overall classification. In this view, statistical measures of an algorithm's performance, the ability of an algorithm to make maximal use of the specific characteristics of the data, and the whatever general parameters are used to evaluate the overall performance of the system (probability of classification, and false alarm per frame) must all be taken into account when evaluating the system. In addition, the models can be used to establish the requirements of the database in terms of data collection and organization, with the end goal of generating databases of sensor data which are increasingly representative of the real world. Thus, sensor/algorithm systems are the best source of information on how to improve themselves. The proposed framework for a computational theory of sensors provides a firm basis for a thorough understanding of the problems involved.

2. Method

In order to achieve the analysis and performance evaluation system described above, it is necessary to divide the problem into more manageable subproblems. The two major aspects of the problem are:

1. the specification of sensors and algorithms which transform sensor data, and
2. the inference of properties of proposed configurations of sensor/algorithm systems.

In this section we discuss our solution to these two problems.

2.1. Logical Sensor Specification

Multi-sensor systems require a coherent and efficient treatment of the information provided by the various sensors. We have proposed elsewhere a framework, the Logical Sensor Specification System, in which the sensors can be abstractly defined in terms of computational processes operating on the output from other sensors [5, 4]. Various properties of such an organization have been investigated, and two implementations have been described.

The principal motivations for logical sensor specification are:

* benefits of data abstraction: the specification of a sensor is separated from its implementation. The multi-sensor system is then much more portable in that the specifications remain the same over a wide range of implementations. Moreover, alternative mechanisms can be specified to produce the same sensor information but perhaps with different precision or at different rates. Thus, several dimensions of sensor granularity can be defined. Further, the stress on modularity not only contributes to intellectual manageability but is also an essential component of the system's reconfigurable nature. The inherent hierarchical structuring of logical sensors further aids system development.

* availability of smart sensors: the lowering cost of hardware combined with developing methodologies for the transformation from high level algorithmic languages to silicon have made possible a system view in which hardware/software divisions are transparent. It is now possible to incorporate fairly complex algorithms directly into hardware. Thus, the substitution of hardware for software (and vice versa) should be transparent above the implementation level.

2.1.1. Logical Sensors

We have briefly touched on the role of logical sensors above. We now formally define logical sensors.

A **logical sensor** is defined in terms of four parts:

1. A **logical sensor name**. This is used to uniquely identify the logical sensor.

2. A **characteristic output vector**. This is basically a vector of types which serves as a description of the output vectors that will be produced by the logical sensor. Thus, the output of a logical sensor is a set (or stream) of

vectors, each of which is of the type declared by that logical sensor's characteristic output vector. The type may be any standard type (e.g., real, integer), a user generated type, or a well-defined subrange of either. When an output vector is of the type declared by a characteristic output vector (i.e., the cross product of the vector element types), we say that the output vector is an "instantiation" of that characteristic output vector.

3. A **selector** whose inputs are alternate subnets and an acceptance test name. The role of the selector is to detect failure of an alternate and switch to a different alternate. If switching cannot be done, the selector reports failure of the logical sensor.

4. **Alternate Subnets**. This is a list of one or more alternate ways in which to obtain data with the same characteristic output vector. Hence, each alternate subnet is equivalent, with regard to type, to all other alternate subnets in the list, and can serve as backups in case of failure. Each alternate subnet in the list is itself composed of:

   * A set of **input sources**. Each element of the set must either be itself a logical sensor, or the empty set (null). Allowing null input permits **physical** sensors, which have only an associated program (the device driver), to be described as a logical sensor, thereby permitting uniformity of sensor treatment.

   * A **computation unit** over the input sources. Currently such computation units are software programs, but in the future, hardware units may also be used. In some cases, a special "do-nothing" computation-unit may be used. We refer to this unit as PASS.

A logical sensor can be viewed as a network composed of sub-networks which are themselves logical sensors. Communication within a network is controlled via the flow of data from one sub-network to another. Hence, such networks are data flow networks.

2.1.2. Implementation

We currently have two implementations of the logical sensor specification language running: a C version (called C-LSS) running under UNIX [3], and a functional language version (called FUN-LSS) [4]. The C version produces a shell script from the specification, while FUN-LSS generates code for a special functional programming language (FEL). FUN-LSS provides a logical sensor specification interface for the user and maintains a database of s-expressions which represents the logical sensor definitions (see Figure 1).

We have defined a Logical Sensor Specification Language as a framework facilitating efficient and coherent treatment of information provided in multi-sensor systems. In

```
                    user
                     ^
                     |
                     v
.---------------------------------------.
| logical sensor specification interface |
'---------------------------------------'
                     ^
                     |
                     v
.---------------------------------------.
| logical and physical sensor database   |
'---------------------------------------'
```

Figure 1. The Logical Sensor System Interface

addition to the issues raised when considering the language implementation itself, various extensions have been suggested. In particular, we have implemented:

* A Logical Sensor Specification Language compiler.

* General fault-tolerance features such as:

  1. A mechanism for detecting two types of sensor failure.

  2. A technique by which switching to an alternate subnet is accomplished.

* A database of physical sensors.

2.2. An Expert System

The knowledge which must be represented depends primarily on the problem to be solved. We have considered three application domains:

* Sensor hardware configuration and design constraints,

* The pixel classification problem, and

* Feature detection.

Concerning the first of these, there are many factors which come into play when a remote sensing system is being configured. For example, the user must specify the signal-to-noise performance which is expected of the system. This index is related to the velocity at which the sensor will be flown, as well as its altitude, field of view and the number of detectors involved. Once these are determined, however, it is possible to choose the spectral channel bandwidth so as to achieve an acceptable signal-to-noise ratio.

In the second example application, consider the classification of corn. If it is necessary to distinguish corn from soybeans, then it is necessary to have data taken in the 1.3, 1.5 to 1.8, and 2.1 to 2.3 micrometer wavelengths. Moreover, seasonal variation must be taken into account, and there must be an algorithm which classifies corn or can be appropriately parameterized to classify corn.

Let's look at a more detailed example of the third application area. Suppose that we wish to know given the current set of sensors on the one hand and a set of algorithms for edge detection on the other, if it is possible to detect edges with a certain absolute resolution in the placement of the edges. Furthermore, suppose that the following two sensors are defined (we are using HPRL, a Heuristic Programming and Representation Language made available to us by HP [6]):

```
***
***   show the sensor frames
***
(M-7 (AKO ($VALUE (AIRBORNE)))
     (BANDS
          ($VALUE (12)))
     (RANGE
          ($VALUE ((0.4 0.9) (UNITS: MICROMETERS))))
     (FOV ($VALUE (90 (DEGREES: !+- 45 FROM NADIR))))
     (THERMAL-RESOLUTION
          ($VALUE (0.1 (UNITS: DEGREES C))))
     (REFLECTANCE-RESOLUTION
          ($VALUE (1 (PERCENT:))))
     (REFERENCE-PORTS
          ($VALUE (5 (RADIATION:))))
     (COLLECTOR-OPTICS
          ($VALUE (12.25 (DIAMETER: CM))))
     (SCANS-PER-SECOND
          ($VALUE (100)
                  (60)))
     (ELECTRONIC-BANDWIDTH
          ($VALUE (90 (KHZ: FROM DIRECT CURRENT)))))
```

```
NIL

(M-7 (AKO AIRBORNE)
     (BANDS 12)
     (RANGE (0.4 0.9))
     (FOV 90)
     (THERMAL-RESOLUTION
          0.1)
     (REFLECTANCE-RESOLUTION
          1)
     (REFERENCE-PORTS
          5)
     (COLLECTOR-OPTICS
          12.25)
     (SCANS-PER-SECOND
          100
          60)
     (ELECTRONIC-BANDWIDTH
          90))

NIL

(LANDSAT-1 (AKO ($VALUE (LANDSAT)))
           (BANDS
                ($VALUE (4)))
           (RANGE
                ($VALUE ((0.8 1.1) (UNITS: MICROMETERS))
                        ((0.7 0.8) (UNITS: MICROMETERS))
                        ((0.6 0.7) (UNITS: MICROMETERS))
                        ((0.5 0.6) (UNITS: MICROMETERS))))
           (FOV ($VALUE (11.56 (DEGREES:))))
           (PIXEL-SPATIAL-RESOLUTION
                ($VALUE (80 (UNITS: METERS SQUARED))))
           (REFLECTANCE-RESOLUTION
                ($VALUE (8 (BITS:))))
           (REFERENCE-PORTS
                ($VALUE (2)
                        (1)))
           (COLLECTOR-OPTICS
                ($VALUE (22.8 (CM:))))
           (SCANS-PER-SECOND
                ($VALUE (13.65)))
           (ELECTRONIC-BANDWIDTH
                ($VALUE (42.3))))

NIL
```

```
(LANDSAT-1 (AKO LANDSAT)
           (BANDS 4)
           (RANGE (0.8 1.1)
                  (0.7 0.8)
                  (0.6 0.7)
                  (0.5 0.6))
           (FOV 11.56)
           (PIXEL-SPATIAL-RESOLUTION
                80)
           (REFLECTANCE-RESOLUTION
                8)
           (REFERENCE-PORTS
                2
                1)
           (COLLECTOR-OPTICS
                22.8)
           (SCANS-PER-SECOND
                13.65)
           (ELECTRONIC-BANDWIDTH
                42.3))

NIL
```

Finally, suppose that the knowledge of the available edge detection algorithms is:

```
***
***   show the edge detector frames
***
(PREWITT (AKO ($VALUE (EDGE-DETECTOR)))
         (INPUT
              ($VALUE (GRAY-SCALE)
                      (IMAGES)))
         (PROCESS
              ($PARMS (THRESHOLD)
                      (QUANTIZATION)
                      (WINDOW-SIZE))
              ($CODE (^PREWITT-EDGE-PROCEDURE)))
         (COV ($VALUE (THETA)))
         (RESOLUTION
              ($VALUE (1.0 (ACCURACY: PIXEL / SUB-PIXEL)))))

NIL

(LAPLACIAN (AKO ($VALUE (EDGE-DETECTOR)))
           (INPUT
                ($VALUE (GRAY-SCALE)
                        (IMAGES)))
           (PROCESS
                ($PARMS (THRESHOLD)
                        (QUANTIZATION)
                        (WINDOW-SIZE))
                ($CODE (^LAPLACIAN-EDGE-PROCEDURE)))
           (COV ($VALUE (RHO)))
           (RESOLUTION
                ($VALUE (1.0 (ACCURACY: PIXEL / SUB-PIXEL)))))

NIL

(TRIENDL (AKO ($VALUE (EDGE-DETECTOR)))
         (INPUT
              ($VALUE (GRAY-SCALE)
                      (IMAGES)))
         (PROCESS
              ($PARMS (THRESHOLD)
                      (QUANTIZATION)
                      (WINDOW-SIZE))
              ($CODE (^TRIENDL-EDGE-PROCEDURE)))
         (COV ($VALUE (RHO)
                      (THETA)
                      (R)))
         (RESOLUTION
              ($VALUE (0.0039 (ACCURACY: PIXEL / SUB-PIXEL)))))

NIL

(PREWITT (AKO EDGE-DETECTOR)
         (INPUT GRAY-SCALE
                IMAGES)
         (COV THETA)
         (RESOLUTION 1.0))

NIL

(LAPLACIAN (AKO EDGE-DETECTOR)
           (INPUT GRAY-SCALE
                  IMAGES)
           (COV RHO)
           (RESOLUTION 1.0))

NIL
```

```
(TRIENDL (AKO EDGE-DETECTOR)
         (INPUT GRAY-SCALE
                IMAGES)
         (COV RHO
              THETA
              R)
         (RESOLUTION 0.0039))

NIL
```

Then if we want to find out whether there is any sensor/algorithm combination which can detect edges at an 80 meter resolution, we obtain:

```
***
***  solve for minimum sub-pixel resolution of 80 meters
***
(solve-all '(?x sub-pixel-resolution 80))

Using Find-Sensor-Algorithm-Pair:
Landsat-1 combined with the prewitt edge detector satisfies the sub-pixel
  constraint of 80 meters.


Using Find-Sensor-Algorithm-Pair:
Landsat-1 combined with the triendl edge detector satisfies the sub-pixel
  constraint of 80 meters.


Using Find-Sensor-Algorithm-Pair:
Landsat-1 combined with the laplacian edge detector satisfies the
  sub-pixel constraint of 80 meters.
```

Upon asking about a 1 meter resolution, we get:

```
***
***  solve for minimum sub-pixel resolution of 1 meter
***
(solve-all '(?x sub-pixel-resolution 1))

Using Find-Sensor-Algorithm-Pair:
Landsat-1 combined with the triendl edge detector satisfies the sub-pixel
  constraint of 1 meters.
```

Finally, at 0.005 meter resolution, we learn that:

```
***
***  solve for minimum sub-pixel resolution of .005 meters
***
```

```
(solve-all '(?x sub-pixel-resolution .005))
NIL
```

Thus, no combination was found for detecting edges at the given resolution. Even from this simple example, it is possible to see the further implications to a more complete set of information.

3. Conclusions

Our primary goal is to produce a system which makes it possible to:

1. Characterize physical sensors in terms of their domains of application, their principles of operation, and their output,

2. Specify algorithms in such a way that the transformations they perform on the input data are well-defined, and

3. Evaluate a given sensor configuration in terms of the sensors involved and the effects of the particular algorithms on the sensed data; this permits the selection of a sensing system optimized according to some user-defined criteria.

The framework based on the interaction of sensors and algorithms allows the determination of the following critical information:

* How well an algorithm is able to take into consideration the characteristics of the sensor.

* How well an algorithm can be implemented and other characteristics of the algorithm itself such as space and time computational complexity, concurrency, modularity, parallelism and the adaptability to the other types of sensors in a multi-sensor environment.

* How well the algorithm is able to predict performance. The performance is verified on a sample data base, and is based on quantitative figures of merit.

We believe that the logical sensor methodology can provide the basis for a computational theory of sensors and that when combined with an appropriate inferencing system makes these goals possible.

References

[1] Bhanu, Bir.
Evaluation of Automatic Target Recognition Algorithms.
In *Proceedings of the SPIE West '83*. August, 1983.

[2] Dorrough, D.C., V.F. Pizzurro, B. Bhanu, and J.R. Pasek.
A Multi-sensor, Multi-mode Tracker Approach to Missile Ship Targeting.
In *1982 Tri-Service Workshop on Missile Ship Targeting*. August, 1982.

[3] Henderson, T.C., E. Shilcrat and C. Hansen.
A Fault Tolerant Sensor Scheme.
Computer Science UUCS 83-003, University of Utah, November, 1983.

[4] Henderson, T.C. and E. Shilcrat.
Logical Sensor Systems.
Journal of Robotic Systems 1(2):169-193, 1984.

[5] Henderson, T.C. and Wu So Fai.
MKS: A Multi-sensor Kernel System.
IEEE Transactions on Systems, Man, and Cyberbetics SMC-14(5):784-791, September/October, 1984.

[6] Lanam, D., R. Letsinger, S. Rosenberg, P. Huyn, and M. Lemon.
Guide to the Heuristic Programming and Representation Language Part 1: Frames.
Technical Report AT-MEMO-83-3, Hewlett-Packard, Application Technology Laboratory, Computer Research Center, 1501 Page Mill Road, Palo Alto, CA 94304, August, 1983.

[7] Nagao, M. and T. Matsuyama.
Advanced Applications in Pattern Recognition. Volume : *A Structural Analysis of Complex Aerial Photographs.*
Plenum Press, New York, 1980.

[8] Tsotsos, J.C.
A Framework for Visual Motion Understanding.
Computer Systems Research Group CSGR-114, University of Toronto, June, 1980.

Table of Contents