Proceedings of the SPIE International Conference on Architectures, Networks, and Intelligent Systems for Manufacturing Integration
Pittsburgh, PA • Oct. 13-17, 1997

An Open Architecture for Intelligent Multisensor Integration in Industrial Applications

Michael D. Naish and Elizabeth A. Croft

Industrial Automation Laboratory
Department of Mechanical Engineering
University of British Columbia
Vancouver, B.C. V6T 1Z4

ABSTRACT

An open architecture framework for intelligent multisensor integration in an industrial environment is being developed. This framework allows for the computational evaluation and understanding of sensor uncertainty and data validity through the comparison of sensor data in a common format.

A logical sensor model is used to represent both real and abstract sensors within the architecture. This allows for the unobtrusive addition or replacement of sensors. All logical sensor outputs are accompanied by a corresponding confidence level. These confidences are used to dynamically allocate valid sensor readings for use by higher-level sensors.

Sensory information is passed to an inference engine which uses user-selectable and adjustable fuzzy logic and/or neural network modules to provide the required decision making intelligence. This architecture may be applied to a broad range of industrial applications, especially those involving non-uniform product grading.

Keywords: sensor integration, open architecture, industrial applications, object representation

1. INTRODUCTION

1.1. Background

The use of intelligent systems and sensor technologies, including machine vision, can be applied to many industrial processes. However, many intelligent sensor systems, while successful from the research perspective, are still too slow to be of practical use. This is problematic, since industry users expect computer systems to offer not only improved quality, repeatability, and reliability but also to maintain and, preferably, increase production speeds.

Recent application examples of industrial sensor integration systems include poultry grading,[1] material surface inspection,[2] printed circuit board inspection,[3] catfish processing,[4] produce classification,[5] and herring roe grading.[6] In many of these applications, ad-hoc methods are used to develop a sensor-based system to monitor the process. Such systems tend to be difficult to understand, maintain, and upgrade. This also is a problem for industrial users whose requirements in terms of speed, feature recognition, accuracy, and other process monitoring parameters invariably change over time.

In many industrial applications, grading, inspection, and process control tasks occur within the controlled environment of a plant. In most cases, the generic recognition problems considered by machine vision researchers are greatly simplified by a priori information. This information encompasses both the background against which objects must be segmented, and knowledge about the objects themselves. Typical production arrangements involve the use of conveyor belts which provide physical separation between objects as the product moves along. This separation eliminates the need for algorithms which perform well when objects are occluded; such algorithms are typically more

Further author information (Send correspondence to E.A.C.) –
E.A.C.: Email: ecroft@mech.ubc.ca; Telephone: 604-822-6614; Fax: 604-822-2403
M.D.N.: Email: naish@mech.ubc.ca; Telephone: 604-822-3147; Fax: 604-822-2403

computationally expensive. In addition, structured and predictable lighting is possible, further simplifying the object recognition task by ensuring that objects appear under the same intensity of light and shadow field.

The industrial context, particularly in the case of product grading and inspection, does not typically require the planning and execution of complex actions. Instead, the sensing of a particular object will likely correspond to a single well defined action. For example, a product exhibiting a particular defect may be ejected into the appropriate bin, while "good" products are simply allowed to continue down the conveyor to a packaging station. Once the defective product is identified, the desired action may be as simple as actuating a valve or solenoid at the appropriate moment.

1.2. Objectives

To address the industry needs outlined above, a new, open architecture for intelligent multisensor integration in an industrial environment is being developed. This framework allows for the computational evaluation and understanding of sensor uncertainty and data validity through the comparison of sensor data in a common format. The objectives of this architecture are as follows:

1. To provide a flexible, easily configurable, and open architecture which serves as a robust platform for intelligent industrial sensing applications.
2. To specify an encapsulation of physical devices and processing algorithms.
3. The specification of a data representation scheme which allows for the quantification of deviations from an ideal model. Such a representation will facilitate the representation of non-uniform objects and allow the comparison of sensor data from diverse sources.
4. To provide a robust exception handling mechanism to ensure the reliability of an implementation of this architecture.
5. The architecture should be applicable to a broad range of industrial applications, especially those involving non-uniform product grading.

1.3. System Overview

The multisensor integration architecture described herein is intended to be open, flexible, and easily configured to a variety of applications within an industrial setting. As many industrial processes involve the evaluation and inspection of products, vision systems offer the primary source of information.

The architecture is object oriented in nature, using *logical sensors* to encapsulate physical sensors and processing algorithms (see Section 2.1 below). The logical sensor hierarchy orders sensor data in a bottom-up manner. The raw data collected by the physical sensors is processed through logical sensors to produce high-level representations of sensed objects and features. While this approach may initially be slower than an ad-hoc implementation, where the required data is extracted directly from the raw sensor data, it offers considerable flexibility. High-level tasks such as non-uniform product grading may be implemented without regard to the specific sensing devices. The low-level physical sensors and low-level data processing routines are transparent. This allows for the addition, removal, and replacement of sensors within the architecture with the minimum amount of disturbance to the overall system. The architecture is illustrated in Figure 1.

The information extracted by the logical sensors is validated before it is passed to the inference engine. Any readings which do not satisfy predetermined constraints, or are in conflict with data from another sensor, invoke the exception handling mechanism, as described in Section 3.

For non-uniform product grading, the representation of objects must allow for the quantification of deviations from an ideal model. The proposed data representation, based upon work by Tomita and Tsuji,[7] addresses this problem by representing objects in terms of user specified features relevant to the grading task, in addition to available 'crisp' data. Object representation and extraction are outlined in Section 4. This high level of data abstraction permits the comparison of sensor information from diverse sources. The features extracted using this representation are then

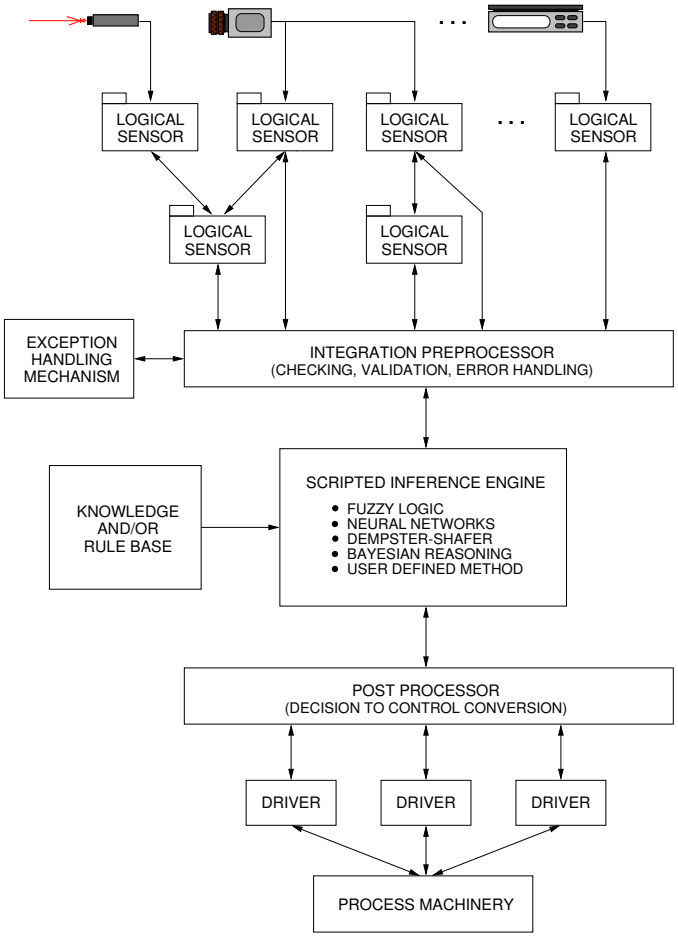

Figure 1. Overall system architecture.

used for discordance based identification and classification. This technique concentrates on distinguishing features to perform object recognition.

The inference engine, described in Section 5, uses sensed objects and features as input to a knowledge and/or rule base which then provides a control decision or directive. Configuration of the architecture is an interactive process. A graphical user interface enables the user to select and configure logical sensors from a library. Links are created to indicate data and control flow. The system is trained by the user; object models, as described in Section 4.1, are constructed using data available from the logical sensors.

2. PERCEPTION AND PREPROCESSING

In this section, the building blocks of the architecture, the logical sensors, are defined and described. As well, the methodology for representing the uncertainty sensed by these devices is outlined.

2.1. Logical Sensors

A logical sensor is an abstract definition for a sensor. Logical sensors were first defined by Henderson and Shilcrat[8] and later broadened to include a control mechanism by Henderson et al.[9] This definition provides a uniform framework for multisensor integration by separating physical sensors from their functional use within a system. One advantage of this definition is that it allows for the simple addition or replacement of sensors. Figure 2 illustrates a logical sensor.

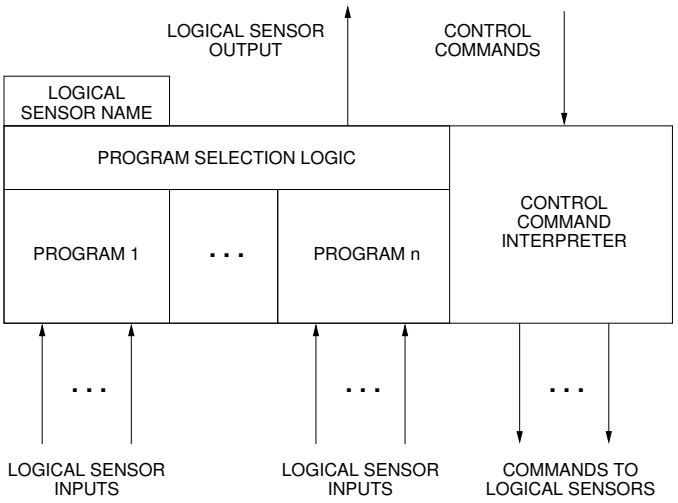

Figure 2. Basic components of a logical sensor.[9]

The logical sensor model provides a control structure which allows for the adjustment of sensors in response to changing conditions. Control commands may be generated from the higher-level logical sensors or the integration preprocessor. The control command inputs may consist of both commands necessary to control the logical sensor and commands that are just passing through to other sensors lower in the hierarchy.

The program selector within each logical sensor monitors the logical sensor inputs, considers control commands from higher-level sensors, and selects the appropriate program. To higher level sensors, each antecedent logical sensor appears as a single entity with a single output, regardless of the scope of *its* antecedents.

Using this definition, physical sensors such as load cells, thermocouples, cameras, and lasers may be represented as logical sensors. In general, a logical sensor reading may be derived from multiple physical readings and there may be a choice of several different algorithms to perform the processing. In this way, "sensors," such as a line detector, which do not physically exist may be made available to the user. Output from a variety of logical sensors may be combined to extract complex features. Physical sensors may be replaced or added without disturbing the entire system – only the associated logical sensor need change.

2.2. Sensing Inaccuracies

Grading and inspection tasks rely upon various sensors to obtain information about the objects under consideration. Accurate decisions require that the sensed information be valid and robust. There are two factors which may cause the sensor system to present data which is inaccurate. The first is uncertainty which may arise due to device limitations (resolution, speed, etc.) or noise. The uncertainty associated with each sensor and algorithm may be characterized but cannot be reduced on an individual basis. Validation of data through sensor integration provides one mechanism by which this uncertainty may be represented, and collaboratively reduced.

Systematic errors are those related to the failure of a particular device to perform within specifications. These errors are not random but rather are the result of improper calibration, sensor malfunctions, or environmental changes. The system checks for these device errors and, if found, attempts to correct the cause of the error. The exception and error handling mechanism is outlined in Section 3.

2.3. Uncertainty Representation

Each logical sensor output is accompanied by a confidence measure. This measure is determined within the logical sensor. It is representative of the known operational characteristics of a physical sensor, or a measure determined by a processing algorithm. The measure is a real valued number between the range 0 and 1. A measure of 0 indicates no confidence in sensor output – 100% uncertainty – while a measure of 1 indicates complete confidence in the output.

2.4. Preprocessing

The preprocessor serves to coordinate sensor integration and exception handling activities. All logical sensor outputs pass through the integration preprocessor before use in the inference engine.

Discordance-based sensor fusion, based upon a cognitive science model proposed by Bower[10] and applied to an architecture for mobile robotics by Murphy,[11] is used to combine information from multiple sensors. There are four fusion modes as follows:

1. Complete sensor unity. In this mode, sensor data is fused without a mechanism for detecting discordances. Sensory information is tightly coupled such that discordances do not arise.
2. Awareness of discordance where recalibration is possible. Here, the discordance between sensors is reconciled by recalibration of the offending sensors.
3. Awareness of discordance where recalibration is not possible. In this case, sensors providing erroneous data are temporarily suppressed.
4. No unity at all. Sensors observe attributes without any spatial correspondence. Here, sensor data is used independently.

Sensor uncertainty is used throughout the integration process. Confidence measures are used for the identification of sensing errors and for the integration of sensor data. Fuzzy logic membership functions, based on a priori sensor output distributions, are used to combine sensor information in a more intelligent and efficient manner than probalistic methods, such as Bayesian reasoning or Dempster-Shafer theory.

As problems are encountered at the logical sensor level, this information is passed to the integration preprocessor. The preprocessor is then responsible for performing the appropriate corrective action. This may involve the adjustment of logical sensor parameters, recalibration of logical sensors, or the reconfiguration of the sensor hierarchy. The removal of malfunctioning sensors from the hierarchy or a reordering of sensors are among reconfiguration possibilities.

3. EXCEPTION AND ERROR HANDLING

Validation occurs both at the logical sensor and the preprocessing levels. Exception handling routines are invoked when data fails to satisfy a predetermined constraint or is in conflict with data from another sensor.

Sensing failures must be handled immediately and quickly to permit the system to continue to operate effectively. Line speeds used for product grading may range from 1 to 30 objects per second. It is unacceptable for products to pass by unevaluated or to slow/stop the line in order to resolve sensing failures.

The exception handling mechanism is based upon the following assumptions:

1. Sensing failures may be caused by sensor malfunctions (power failure, miscalibration, etc.), changes in the environment which adversely affect sensor performance (change in light levels, obscured lens), or errant expectation (expected object not in scene).
2. The exception handling mechanism may not have a complete causal model of all possible sensing failures – such a model is difficult to construct. The consequence of this is that there may be no way to determine the cause of a sensing failure detected as missing, highly uncertain, or conflicting with other data.[12]
3. On-board sensor diagnostics, or the sensor data which led to a failure, may be used by the exception handling mechanism. Additionally, a sensor not in the sensing plan may be used to externally corroborate the functionality of the suspect sensor or to verify that the environmental pre-conditions remain unchanged.
4. In the event that the cause of a sensing failure cannot be identified, the exception handling mechanism will assume an errant expectation.

3.1. Error Classification

Without the availability of a complete causal model, detected errors must be classified so that the appropriate corrective action may be taken. Sensor failures are classified into three types as follows:

1. Sensor malfunctions: This occurs when one or more sensors are malfunctioning. Examples include power failure, impact damage, miscalibration, etc.
2. Environmental change: One or more sensors are not performing properly because the environmental conditions have changed since sensor configuration and calibration.
3. Errant expectation: Sensor performance is poor because the sought object is occluded or lies outside of the sensor's "field of view."

Error classification is accomplished by a generate and test algorithm.[12] The suspect sensors are first identified. An ordered list of possible hypotheses explaining the sensor failure is then generated. Each hypothesis is associated with a test which may be used for verification. These tests are performed in an effort to confirm or deny the proposed hypotheses. This process is repeated until a hypothesis is confirmed.

The generate and test method does not require formal operators for the generation of hypotheses. This allows the system to use a rule-based method to select from a list of candidate hypotheses. Unfortunately, this method can be time consuming if there is a large problem space and all hypotheses must be generated. This disadvantage may be overcome by not constraining the problem space, thereby limiting the number of hypotheses and reducing processing time. Testing is conducted until all tests have been performed or an environmental change has been detected. When the classifier is unable to resolve the cause of the error, the cause is assumed to be an errant expectation.

3.2. Error Recovery

For each error cause, there would ideally be a number of different recovery schemes. From these, the most appropriate would be selected by the exception handling mechanism. To limit the scope the problem and reduce the overall recovery time, a direct one-to-one mapping of error causes to recovery schemes is utilized. A library of cases allows for the instant mapping of error cause to recovery scheme based on the error classification.

Functions are used to repair individual sensors or reconfigure the sensor hierarchy. The sensor parameters are adjusted first – recalibration is accomplished by invoking a predefined sensor calibration routine. If the sensing configuration cannot be repaired through parameter adjustment or recalibration, the sensor hierarchy is altered. The alteration may suppress a particular sensor or remove sensors from the hierarchy by the simple adjustment of pointers.

4. OBJECT EXTRACTION

Objects are extracted through the use of the appropriate logical sensors. The logical sensors which are available to the user depend on upon the types of physical sensors attached to the system. For example, a load cell may utilize a single logical sensor which provides measurement of an object's mass. On the other hand, a CCD camera may have a much larger number of associated logical sensors. The camera itself provides a raw image from which an edge detecting logical sensor can extract edges. Similarly, a line detecting logical sensor can extract straight line segments from the edge detecting logical sensor or a colour logical sensor can extract relevant colour information – all from the same raw image. Then, these sensors may be combined to detect compound objects.

Common logical sensors are available to the user from a library. The associated algorithms are chosen primarily for computational efficiency. Additional logical sensors may be added to the library to address special user needs, add new sensors, or introduce new algorithms. The user chooses logical sensors and/or combinations of logical sensors to achieve the desired level of performance. In general, this will involve a trade-off between accuracy and speed. After selecting the logical sensors to extract the desired information, the associated parameters are adjusted by guided search to achieve the desired results.

Object types are determined by the extraction capabilities of the logical sensors. Various edge detectors, segmentation algorithms, texture identifiers, colour models, and other image processing routines each produce particular object types: lines, regions, textures, colours, etc. These types are used to identify the extracted objects and features.

4.1. Object Representation

Objects are represented in a top-down manner, the root node of the hierarchy representing the overall object. Objects are represented through a combination of distinguishing and disparate features. The tree is built of object nodes which represent a recognizable object or feature. Features which are not always present in a parent node are marked by a *free node tag*.

Object nodes may be linked unconditionally or conditionally. An unconditional link represents a *parent-child* relationship between nodes. Conditional links are used to represent an *exclusive OR* relationship. This is particularly useful for non-uniform objects which may have a number of acceptable deviations. The object representation is discussed further in a previous work.[13]

4.2. Object Recognition

The task of object recognition follows from the object representation. The presence or absence of particular features and the associated object properties are used to classify an object into a particular grade.

There are two approaches which may be taken towards object classification and grading. The first is an extension of the object recognition problem. The entire object is identified based upon the features contained within the object model and extracted by the logical sensors. The identified object will fall into a predetermined classification. Extracted object properties may then be used to further evaluate the object based on attributes such as size, colour, and mass. This information may then be used in the control of automated equipment for the separation of the different classifications.

For complex objects it may be more efficient to define the object models somewhat differently. Instead of attempting to identify the entire object, only distinguishing features are extracted. The features may then used as input to an inference engine. The presence or absence of particular features and the associated object properties may then be used to classify the object into a particular grade. This approach, utilizing disparities between objects, has the advantage of a more compact representation. Recognition proceeds more quickly by keying in on uniquely identifying features.

5. INFERENCE ENGINE

Once the sensory information collected by the logical sensors has been validated, it is passed to the inference engine. Here, based upon the examination of the extracted objects and features, decisions are made regarding the actions to be taken with each object. Unlike mobile robotic applications, there is no sensing plan. In other words, the sensors are not actively manipulated to gather information (i.e. the panning of a camera to look around a room). The industrial context neither requires this nor affords adequate time for such operations.

The inference engine is designed to make use of cognitive-based models such as fuzzy logic and knowledge based systems selected and configured by the user. Other possibilities include feature-based inference techniques such as Bayesian reasoning, the Dempster-Shafer theory of evidence, or artificial neural networks.

User configuration of the inference engine for a particular task proceeds concurrently with the construction of the logical sensor hierarchy as outlined in Section 4. Typically, the user will start with a number of objects to be identified along with a corresponding set of features. A logical sensor hierarchy is then constructed to sense these features. The sensor inputs are used to form the antecedents of the the control decisions to be made in the inference engine. The consequents of these rules are the actual decisions. These are passed from the inference engine to the postprocessor for conversion into action.

The system provides a number of building blocks which may be either used directly or modified by the user for a particular application. These blocks provide a systematic way of constructing the inference mechanism. All aspects of the system are represented, sensors and actuators, so that percepts may be linked to actions. Additionally, the user may construct new building blocks to provide functionality not already provided. Programming is achieved through the use of a scripting language similar to that of MATLAB®.

Fuzzy logic or knowledge based inference requires expert domain knowledge to be supplied by the user. This knowledge is placed into the rule base associated with the inference engine. Contextual knowledge may be used to

weight the contribution of various sensors. For grading and inspection tasks in particular, the expert knowledge available from human inspectors is available to the system designers.

For applications where expert knowledge is less concrete, artificial neural networks may be used to interpret the sensor information and produce control decisions. In this case, the network must be interactively trained to produce the desired results.

6. FUTURE WORK

6.1. Post Processing and Control

Once the inference engine has processed the sensory information and interpreted it, any decisions made must be converted to actions. This involves the conversion of a directive into a plan of action for execution. For example, the decision to place a bruised apple into the "bruised apple bin" must be translated such that the appropriate actuators effect this action at the appropriate time. In the case of food processing, the action is usually conducted immediately as the product moves along a conveyor belt.

One approach to this may be the concept of a *logical actuator*. Control decisions made within the inference engine would be passed to a logical actuator hierarchy where directives are converted into actions. The logical actuators thus serve as an interface between the high-level decision making system and the low-level process machinery. In this sense, a logical actuator is an analogue to a logical sensor. The low-level drivers and actuators are encapsulated by the logical actuator model. The physical actuating devices, or planning algorithms may be altered without affecting the inference engine.

A logical actuator may encapsulate trajectory planning algorithms, abstract actuator subsystems, drivers, or physical actuators. As such control actions are both handled and, in the case of higher-level logical actuators, generated by logical actuators.

6.2. Application

Having determined a suitable model for the representation of the sensors and actuators used within an industrial setting, future work will involve the prototype implementation of the architecture. A potential application is the grading of a non-uniform food product such as herring roe. Prior work undertaken in the UBC Industrial Automation Laboratory in this area[6,14] provides a suitable test bed upon which a multisensor integration system may be built and tested. The prototype implementation will serve to verify the system operation and its performance may be compared to earlier single-sensor implementations.

The use of software agents will be considered in an effort to further increase the openness and flexibility of the system. This is an extension of the object oriented nature of the logical sensor and logical actuator hierarchies. Through the use of these software agents, each sensor, algorithm, controller, actuator, etc. becomes a separate module which may interact with other modules through a specified protocol. Dependencies on particular hardware configurations and software algorithms are further reduced, if not eliminated.

7. SUMMARY

To address the needs of industrial users, who require easily configurable, reliable, and flexible sensor systems, an open architecture for multisensor integration is being developed. Using a logical sensor model to encapsulate physical and abstract sensors, a bottom-up sensor hierarchy is constructed. This provides high-level decision making and inference modules with abstract information, independent from the low-level sensors. Sensor reconfiguration may proceed with minimal impact on the overall system.

A robust exception handling mechanism, which serves to monitor and reconfigure the sensor hierarchy as necessary, ensures that the sensory input available to the inference engine is as accurate as possible. Sensor errors are classified using a generate and test methodology. On-line error classification and recovery is designed such that the problem space is not fully constrained.

The inference engine interprets the sensor input relative to the object models. The object models consist of a connected graph structure which indicates the feature dependencies in a top-down manner. The user configured inference engine utilizes fuzzy logic or neural networks to generate control decisions from the sensory input.

There are many applications which may benefit from this architecture. Examples include the sorting and grading of non-uniform food products such as meat, fish, fruit, and vegetables. In fact, the grading of any product, uniform or not, should be facilitated by this architecture. Other industrial applications which could benefit from sensor technologies, such as machining and manufacturing operations, will be the focus of future applications.

ACKNOWLEDGEMENTS

This work was supported by grants from the *Garfield Weston Foundation*, the *Natural Sciences and Engineering Research Council of Canada*, and the *Gordon M. MacNabb Scholarship Foundation*.

REFERENCES

1. W. Daley, R. Carey, and C. Thompson, "Poulty grading/inspection using color imaging," in *Proceedings of the SPIE - Machine Vision Applications in Industrial Inspection*, vol. 1907, pp. 124–132, SPIE, 1993.
2. J. H. Dan, D. M. Yoon, and M. K. Kang, "Features for automatic surface inspection," in *Proceedings of the SPIE - Machine Vision Applications in Industrial Inspection*, vol. 1907, pp. 114–123, SPIE, 1993.
3. G. Brown, P. Forte, R. Malyan, and P. Barnwell, "Object oriented recognition for automatic inspection," in *Proceedings of the SPIE - Machine Vision Applications in Industrial Inspection II*, vol. 2183, pp. 68–80, SPIE, (San Jose, CA), 1994.
4. P. Jia, M. D. Evans, and S. R. Ghate, "Catfish feature identification via computer vision," *Transactions of the ASAE* **39**, pp. 1923–1931, Sep./Oct. 1996.
5. R. M. Bolle, J. H. Connell, N. Haas, R. Mohan, and G. Taubin, "Veggievision: A produce recognition system," in *Proceedings of the 1996 3rd IEEE Workshop on Applications of Computer Vision*, pp. 244–251, IEEE, (Sarasota, FL), 1996.
6. E. A. Croft, C. W. de Silva, and S. Kurnianto, "Sensor technology integration in an intelligent machine for herring roe grading," *IEEE/ASME Transactions on Mechatronics* **1**, pp. 204–215, Sept. 1996.
7. F. Tomita and S. Tsuji, *Computer Analysis of Visual Textures*, Kluwer Academic Publishers, Norwell, Massachusetts, 1990.
8. T. C. Henderson and E. Shilcrat, "Logical sensor systems," *Journal of Robotic Systems* **1**(2), pp. 169–193, 1984.
9. T. C. Henderson, C. Hansen, and B. Bhanu, "The specification of distributed sensing and control," *Journal of Robotic Systems* **2**(4), pp. 387–396, 1985.
10. T. G. R. Bower, "The evolution of sensory systems," in *Perception: Essays in Honor of James J. Gibson*, R. B. MacLeod and H. L. P. Jr., eds., pp. 141–153, Cornell University Press, Ithaca, NY, 1974.
11. R. R. Murphy, "Biological and cognitive foundations of intelligent sensor fusion," *IEEE Transactions on Systems, Man, and Cybernetics–Part A: Systems and Humans* **26**, pp. 42–51, Jan. 1996.
12. G. T. Chavez and R. R. Murphy, "Exception handling for sensor fusion," in *Proceedings of the SPIE - Signal Processing, Sensor Fusion, and Target Recognition VI*, pp. 142–153, SPIE, (Boston, MA), Sep. 7-10, 1993.
13. M. D. Naish and E. A. Croft, "Data representation and organization for an industrial multisensor integration architecture." to appear in Proceedings of SMC '97.
14. S. Kurnianto, "Design, development, and integration of an automated herring roe grading system," Master's thesis, Department of Mechanical Engineering, University of British Columbia, Vancouver, B.C. V6T 1Z4, June 1997.