MVA '98 IAPR Workshop on Machine Vision Applications, Nov. 17-19, 1998, Makuhari, Chiba, Japan

8—27

Model Management in the System Generating Vision Inspections

Martin Krčmář, Petr Kodl*
ProTyS Ltd. and Rockwell Automation Ltd., Research Center Prague

Abstract

The article presents a possible approach to model management in a knowledge-based system with distributed knowledge. The knowledge based system is designed to automatically generate vision inspections. The proposed model management mechanism extends system reasoning power without increasing system complexity. The model management knowledge is structured similarly to knowledge for inspection synthesis. Consequently, important advantages of knowledge organization like sharing, flexibility, independence, etc. are preserved for both kinds of knowledge.

1 Automatic Vision Programming

Automatic design of industrial vision inspections can be viewed as a computer aided synthesis of vision programs. Systems providing this type of services are designed especially to enable efficient utilization of large vision libraries even by experts from other fields (see [1, 2, 3, 4]).

Comparing to general vision programs, industrial vision inspections exhibit some specialities. Used vision algorithms are relatively simple due to speed requirements. Processed images are usually of good quality because acquisition process can be controlled. On the other hand, vision inspections are executed repeatedly on many images for a long time. Image properties can vary within certain range thus the inspection has to be well adjusted.

Systems generating vision inspections do not build programs from scratch. Instead, they usually compose inspection algorithms from library functions, called vision tools. To produce reliable, well adjusted inspections the systems have to utilize all available information during inspection set up. Beside training images they often retrieve a lot of context information interacting with a user. Models of inspected objects represent important pieces of such information.

Systems for aided inspection design usually can accept CAD object models and set up inspections accordingly. However, they are not able to build and manipulate models if these are not provided externally (see [6, 5]). The CAD models also do not contain some important information, for instance, explicit description of object defects as they occur in training images. Presented model management overcomes some of these disadvantages.

*Address: Americka 22, 120 00 Prague, Czech Republic. E-mail: `martink@rockwell.cz`, `pecold@rockwell.cz`

2 Vision Inspection Synthesis

Our testing environment SYGEVIN (see [7]) is a knowledge-based system that automatically generates vision inspections. It accepts three types of input data:

- specification of vision inspection objectives,
- training images,
- context information.

All these pieces of information are provided by the user. Based on input data, the system generates vision inspection programs. These programs define inspection structure, i.e. sequence of selected vision tools together with data flow, and parameter values for used vision tools. The system is able to generate more inspections if its database allows alternative solutions of given task.

2.1 Reasoning Model

The reasoning for inspection synthesis consists of two subtasks:

- planning
- parameter selection

The *plan reasoning* is carried out as a skeletal planning (see [8]). Given inspection objectives define an inspection goal for which the system selects the most suitable skeletal plan from its database. The skeletal plan itself is again a sequence of goals, i.e. inspection sub-goals. The sub-goals can be both *simple* and *complex*. Simple goals are fulfilled by applying one vision tool while complex goals have to be further decomposed. Skeletal plan based decomposition continues until the plan consists of only simple goals.

The *parameter selection* is carried out as rule based reasoning. On contrary to planning, which is goal-driven, parameter selection is data-driven. Parameters are usually optimized using trial-and-error approach. Constraints on vision tool outputs are determined and then parameter values are selected to satisfy them. For instance, optimizing a gage tool the system identifies range where the edge should occur and then adjusts correlation kernel.

System reasoning is implemented using AND/OR schemes. Scheme structure reflects skeletal planning hierarchy. Scheme nodes encapsulate data driven decision making. It is possible to distinguish three node types:

AND nodes, called *Plans*, correspond to skeletal plans,

OR nodes, called *Abstract Operators*, correspond to complex goals,

leaf nodes, called *IPP Operators* [1], correspond to simple goals.

Decision making knowledge is represented by rules. It is distributed among the scheme nodes. Knowledge is organized such that different types of nodes make different decisions. *OR nodes* select the most promising plans from connected AND nodes. They define sequences in which alternative plans are examined. *Leaf nodes* select parameter values for vision tools. They address parameters that can be optimized at this level. *AND nodes* select values for parameters that have to be coordinated over several tools. For instance, detecting object position using gage tools the tool locations have to be selected for all gage tools at once.

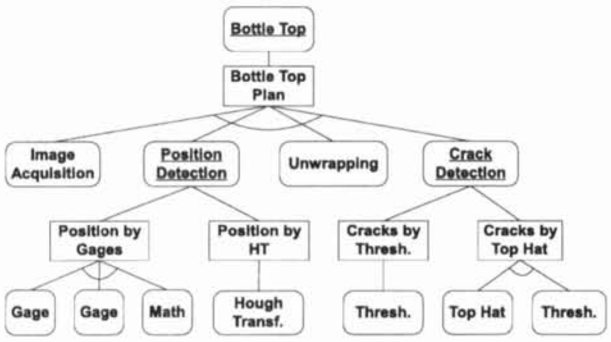

Figure 1: Bottle Top Inspection Tree: rectangles = plans (AND nodes), rounded rectangles = operators (OR nodes - underlined text).

The inspection is generated by traversing the AND/OR scheme. This process forms an inspection AND/OR tree (see Figure 1). Within this tree, the system allows to propagate decisions only downward. For instance, parameter values selected in AND nodes are propagated to corresponding leaf nodes. It is not possible to propagate any decision into upper levels. This restriction imposes quite strong structure to the system knowledge and increases flexibility. Nodes can be plugged into other inspections without modifying their internal knowledge. If upward decision propagation was supported, nodes would have to understand context in which they are used to be able to send up useful decisions. However, this context differs for different inspections. Consequently, internal node knowledge would have to be modified with each new inspection the node was used in. This approach would result in an interrelated unmaintainable database.

[1]IPP stands for Image Processing Procedure.

2.2 Bottle Top Inspection

Let us show inspection synthesis on a simple example of the Bottle Top Inspection. The objective is to detect cracks on the glass top surface. The inspection consists of four basic steps:

1. Image Acquisition,
2. Bottle Top Position Detection,
3. Bottle Top Unwrapping,
4. Crack Detection.

Processed images are shown in the Figure 2. Bottle position can be determined either by the Hough transformation looking for a bright circle or by two gage tools detecting a bottle edge. The cracks can be detected either by simple thresholding or by the morphological Top Hat operator. The inspection AND/OR tree is depicted on the Figure 1. Based on this tree, it is possible to generate at most four alternative solutions.

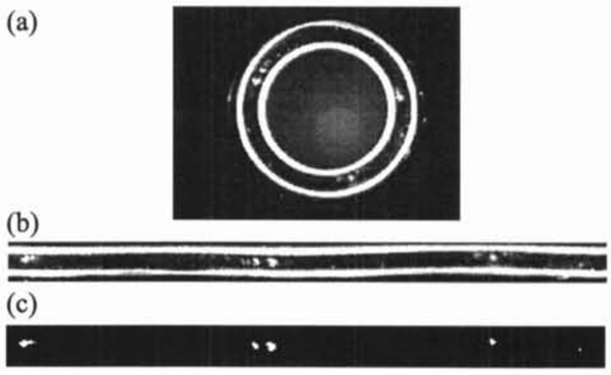

Figure 2: Bottle Top Inspection Images: (a) input, (b) unwrapped, (c) segmented cracks.

Inspection nodes contain various decision-making knowledge, for example: The "Position Detection" operator selects plans according to speed requirements, the "Thresholding" operator optimizes threshold value, and the "Position by Gages" plan places gage tools.

3 Model Management

Decision making carried out in both operator and plan nodes usually requires rich context information to derive correct conclusions. Context information is especially crucial for parameter optimization. It often requires knowledge of expected tool outputs. For instance, threshold for image binarization can be properly set only if requested binary image is known a priori. It is also the only case when binarization performance can be evaluated. Majority of such context information can be directly or indirectly derived from models of inspected objects. Therefore, explicit support of object models can significantly improve reasoning for inspection synthesis.

The model is considered to be any piece of information that relates to an inspected object and that can be utilized for decision making. The model usually describes important object features. It is parameterized by training data. One object model can be created for each training image if necessary. For instance, the sufficient object model for the Bottle Top inspection consists of two circles and one binary image that identify bottle top inner and outer rims and crack pixels respectively, see Figure 3. This structure contains all information necessary for selecting parameter values for both position detection and crack detection.

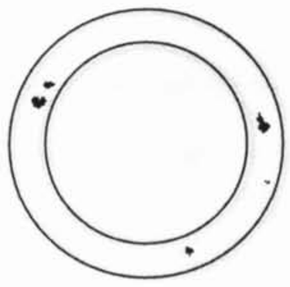

Figure 3: Bottle Top Model: two circles and binary image (on a background of a grayed bottle top).

Since the object model contains important information for decision making it has to be shared by all inspection operators. However, the model cannot be passed among them directly. Model structure depends on inspected objects, i.e. on inspection context. On the other hand, inspection operators should be context independent to keep database flexible. For instance, it does not make sense for the Gage operator to understand complete models of various objects. What the operator only needs to know is a range where an edge occurs within the training image. Based on this information, the Gage operator can adjust the Gage tool to that specific edge regardless of the inspection it is a part of.

To keep database flexible the operators are designed to understand and manipulate predefined model types. The model types correspond to tasks the operators are designed to deal with. For instance, the thresholding operator accepts binary image, the gage operator accepts a range of edge occurrence, the Hough transformation accepts description of a searched circle, etc. Since one inspection tree can contain operators working with different model types, the model has to be transformed when passed among operators.

The proposed model management is designed such that the system does not rely on externally provided models. It is able to create model by itself, hence working partly as an image understanding system. However, it strongly relies on interaction with the user, thus model development is just semi-automatic.

3.1 Model Management Knowledge

The model is built as the system traverses the inspection AND/OR tree. Model development knowledge is distributed among nodes like knowledge for inspection synthesis. This distribution stems from two reasons:

1. equivalence of root and branch operators,
2. sharing of model development knowledge.

Ad *(1)*, the system does not distinguish any specific class of root nodes. Any operator node can become a root node which allows the user to program both the whole inspections and any of their parts. The root node is not provided with a model therefore no node can rely on externally given one. If the model is not provided but it is crucial for reasoning, the operator has to create it by itself. Ad *(2)*, well structured operators enable sharing of model development knowledge. For instance, the "Detect Position" operator of the Bottle Top inspection searches for two circles, bottle rims, that contain sufficient information for position solving operators. The "Detect Position" operator can be used in other bottle inspections, for example in scuffing evaluation. In such case, the model development knowledge can be utilized too.

The model management knowledge is represented by rules. Tasks accomplished by these rules correspond to a node type:

- *Operators* represent working units. They encapsulate specific image processing. The operators have to determine selected characteristics of inspected objects to be able to properly adjust processing behavior.

- *Plans* link operators together. They administrate sharing of detected object properties (object models).

Plan behavior depends on whether the model is provided or not. Plans enable two ways of model management:

- *top-down:* If the model is provided, plans retrieve model pieces as they are required by plan operators. These pieces are transformed into a suitable form before passing to the operators.
- *bottom-up:* If the model is not provided, plans collect partial models developed by operators, compose them into one model and check its consistency. If consistency is broken the model is corrected and last reasoning is redone.

The top-down approach is suitable if the object model is complex, highly constrained, therefore should be created at once having all information available. The bottom-up approach is suitable if the model is simple, consisting of relatively independent parts, therefore allowing wide sharing of model development knowledge. See Figure 4.

In the Bottle Top example, there are two operators building the model. The "Detect Position" identifies bottle rims and the "Crack Detection" segments cracks. The "Bottle Top Plan" composes these pieces and checks if all cracks are within the circles. Other plans transform partial models into appropriate models for tool operators. For instance, the "Position by Gages" sets gage positions and then calculates intersection of gages with a selected circle. It is the place where the edge should be searched for.

4 Conclusion

The presented model management considerably improves reasoning for inspection synthesis. Model development is driven by synthesis process which allows the system to focus on relevant object properties. The necessary condition of model sharing resulted in predefined model types which, in turn, allows sharing of model development knowledge.

Although inspection is generated top-down, the model can be built in both top-down and bottom-up ways. Both approaches can be combined. It enables distribution of model development that is adequate to solved tasks. Model development in the abstract operators utilizes implicit knowledge given by operator tasks, thus it is more reliable. On the other hand, model development in the IPP operators enables wider sharing of the model knowledge.

The presented model management fits well into system reasoning. It does not increase system complexity while preserving database flexibility. Model management knowledge is well structured like knowledge for inspection synthesis. It stems from correspondence between organization of knowledge for skeletal planning and structure of object models (inspection knowledge as an AND/OR tree - model as a graph/tree structure).

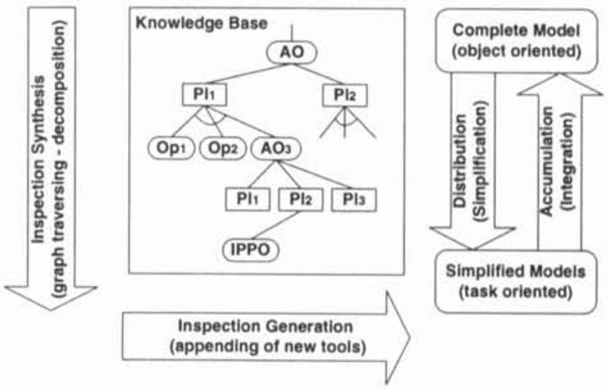
Knowledge Base
AO
Pl1
Pl2
Op1
Op2
AO3
Pl1
Pl2
Pl3
IPPO
Inspection Synthesis (graph traversing - decomposition)
Inspection Generation (appending of new tools)
Complete Model (object oriented)
Distribution (Simplification)
Accumulation (Integration)
Simplified Models (task oriented)

Figure 4: Model Development in a Context of Inspection Synthesis.

References

[1] T. Matsuyama. Expert systems for image processing: Knowledge-based composition of image analysis processes. *CVGIP*, 48:22–49, 1989.

[2] V. Clément and M. Tonnant. A knowledge-based approach to integration of image processing procedures. *CVGIP: Image Understanding*, 57(2):166–184, March 1993.

[3] S. A. Chien. Automated synthesis of image processing procedures for a large-scale image database. In *Proceedings of ICIP'94* , pages 796–800. 1994.

[4] S. Moisan and M. Thonnat. Knowledge-based systems for program supervision. In *1st International Workshop on Knowledge Based Systems for the (re)Use of Program Libraries KBUP'95*, INRIA, pages 4–8, Nov 1995.

[5] R. Bodington. A software environment for the automatic configuration of inspection systems. In *Proceedings of KBUP'95*, pages 99–108.

[6] R. Sablatnig. *A Highly Adaptable Concept for Visual Inspection.* PhD thesis, Technischen Universität Wien, Februar 1997.

[7] M. Krčmář and P. Kodl. System composing vision inspections. In *Engineering of Intelligent Systems EIS'98*, pages 404–410. ICSC Canada, ICSC Academic Press, 1998.

[8] P. R. Cohen and E. A. Fiegenbaum. *Handbook of Artificial Intelligence*, volume III. Pitman, London/New York, 1982.