A Flexible Concept for Automatic Visual Inspection

Robert Sablatnig

Technical University Vienna, Institute of Automation,
Pattern Recognition and Image Processing Group
Treitlstr. 3 / 183-2, A-1040 Vienna, Austria
e-mail: sab@prip.tuwien.ac.at

Abstract

This work aims to show a systematic automated visual inspection concept that speeds up the development of such systems by providing a software reuse and setup concept. A new, highly adaptable concept for visual inspection separates the detection of primitives from the model-based analysis process. This separation is reached by defining a general analysis graph for inspection, containing detail relations that represent detection algorithms. Together with an object specific description, defined in a so called description language, the analysis graph is instantiated. Existing pattern recognition software is re-used in the detection stage and therefore the use of any detection algorithm is possible without changing the analysis. The flexible visual inspection concept is based on a systematic approach, that is able to increase the degree of flexibility and thus to decrease the set-up costs taking into account different type of data, positioning, occlusions and tolerances. The concept can be seen as "recipe" for solving industrial applications, stating at which stage, which kind of decisions have to be made.

The visual inspection of analog display measuring instruments serves as a demonstration of this concept. The flexibility of the concept is demonstrated by testing the analysis process with the description of two instruments (a hygrometer and a clock), which is performed by adapting the analysis graph but without changing the detection algorithms. Since the detection is represented as detail relation in the analysis graph, a change of the detection algorithm is possible without changing the overall analysis. The separation of detection and analysis therefore ensures, that any existing pattern recognition software can be re-used.

1 Introduction

Industry needs **a**utomated **v**isual **i**nspection (AVI) because in the manufacturing process uncertainties like tolerances, defects, relative position and orientation error, etc. exist, which can be resolved by vision sensing [FRE89]. Vision has undergone a development of over 30 years of research (in the computer vision field) and application (in the machine vision field), resulting in a wide range of different algorithms and systems. The problem is how to match the technology to a specific application in an optimal, cost-effective way. From the viewpoint of industry there are three main difficulties for automated visual inspection to be used widely:

- *Cost*: Development costs are still too high; vision software development, in particular makes up a substantial portion of the cost of a new application.
- *Technology*: At present AVI systems require controlled environment and precise positioning, are unable to handle shadows, highlights, and occlusions [CHI92].
- *Ad hoc solutions*: Solutions to ensure robustness are very specific in their application [MAR92]. This approach seems to be the best way to solve a particular problem since such solutions are dealt with case-by-case and tend to be simpler, more compact, and cheaper, but customized techniques for specific problems cannot be duplicated or reused.

The major drawbacks are the high set-up costs resulting from the extensive pre-inspection set-up by experienced operators, hard- and software development costs, labor, and maintenance costs. The key to solving the problem of flexibility is the development of visual inspection systems which are able to inspect a large variety of different objects without or only partly changing the analysis algorithm. In short, there is a requirement for generic visual inspection cells based on the principles of "Soft Automation", in which product modelling is coupled with flexible manufacturing systems and flexible inspection cells which can produce any product with the minimum of design and quality control constraints [TOD88].

A systematic approach ensures that costs are reduced, due to re-usable systems and that less qualified personnel can operate them, with the help of a user interface. As far as speed and technology are concerned, faster hardware is under development (the processing speed doubles every year), so that better algorithms can be used to solve vision problems. At present, however, these systems are not available; there are only prototypes in use and almost no new concepts can be found in the literature. The main reason for this is company politics; developments made by so-called *Machine Vision Companies* are not reported in the literature, since competitors in the market could use these concepts and sell them better. In the academic field only a few researchers are working on systematic AVI, the majority is trying to adapt computer vision algorithms to machine vision applications (see [NEW95]), struggling with system engineering problems [BAT96]. Therefore, work on this topic helps to advance progress in AVI.

This paper shows a new visual inspection concept that speeds up the development of AVI systems by providing a concept for software re-use and systematic set-up, adding a higher degree of flexibility to the inspection system. This flexibility decreases the set-up costs by taking into account different types of data, positioning, occlusions and tolerances. The speed constraint in terms of hardware will not be discussed, since processing capability is still increasing and price is decreasing. The global concept is explained in the following section. Following a case study of the concept for different types of analog display instruments in section 3 the concept is discussed in the conclusion.

2 Visual Inspection Concept

The AVI system design cost has already been reduced by the development of libraries of image processing algorithms (e.g. Matrox Image Library [MAT96]) or interactive image processing systems (like Khoros [KHO95], KBVision [AME95], and Matrox Inspector [MAT96]), which allow rapid prototyping and the re-use of algorithms. However, it is unlikely that the common inspection system user will have the relevant image processing expertise to be able to set up an inspection system. If image processing systems are to be adopted and used for inspection, it is essential to reduce the expertise required in the configuration of the inspection system [BOD95].

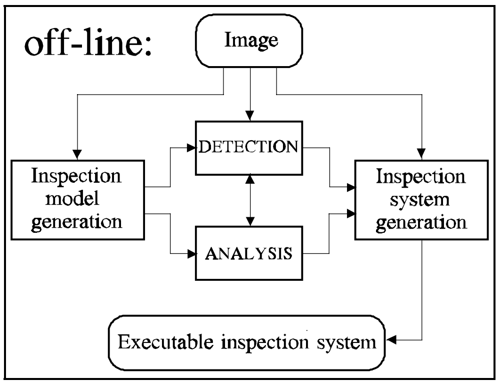

off-line:
Image
DETECTION
Inspection model generation
Inspection system generation
ANALYSIS
Executable inspection system

Figure 1 General inspection system concept

One solution in designing a flexible visual inspection system lies in the separation of the application-independent feature *detection* from the application-dependent *analysis*, forming the model-based AVI system [SAB95,SAB96]. Figure 1 shows the *off-line* part of the proposed concept (rectangles indicate processes, rounded rectangles data). This part is called *off-line* since it is not performed at the speed of the production line. The generated executable inspection system is used on-line, i.e., it works within the production line flow.

2.1 Inspection Model Generation

All visual inspection systems use a priori knowledge to perform the inspection. Therefore, the first step in the set-up of an inspection system is the off-line generation of the inspection model. The goal of the inspection model generation is to provide a description of complex objects using a small set of simple primitives and relations (or structural rules). Due to the structural rules this approach is referred to as structural or syntactic pattern recognition approach [FU82, BUN92]. Syntactic methods use the internal object structure as an analysis element, based on the fact that an object can be described recursively from simple primitives through its structure.

In order to provide a general concept, primitive types are not limited or constrained. The application type (binary or intensity images,from cameras, X-ray, ultrasonic, range sensors and other), data type (sparse, dense, noisy) is thus application dependent. Generally, the following constraints for choosing primitives have to be considered [DAR88,SAB96b]:

- Primitives should have a unique name and there should be a finite number of primitives.
- Primitives should be independent, i.e. not defined in terms of each other.
- The set of primitives should be compact. No subset should be replaceable by smaller sets.
- Primitives should be comprehensive. They should be basic pattern elements to provide compact but adequate description of patterns in terms of specified structural relation.

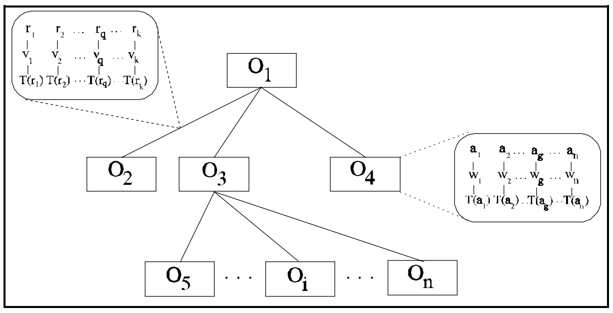
The object structure (shape primitives and properties) is represented in a description language consisting of a graph structure in which nodes represent primitives and arcs relations between primitives. A priori information concerning the quality standard (e.g. manufacturing and detection tolerances) are also part of the model. Modeling can be interpreted as a syntactic pattern recognition approach in which primitives are transformed into a vocabulary and relations are transformed into a grammar [FU82]. This concept can be seen as an application of semantic networks [DAR88], since semantic networks are labeled, directed graphs where nodes represent objects, sub-objects, or shape primitives and arcs represent relations between them. A set of attributes that describe different object features is attached to each node; a set of attributes that describe different properties is attached to each arc. Once the object is transformed into this representation operations for recognition, verification, and inspection can be executed on this graph structure. The advantage of a description language lies in the uniqueness of representation, different objects result in different descriptions.

Formally, the description language is a graph $G=<\mathcal{O},\mathcal{R}>$, where $\mathcal{O}=\{\ m \mid 1 \leq m \leq n\ \}$ denotes the set of nodes and $\mathcal{R}=\{<c,d> \mid c,d \in \mathcal{O}\}$ the set of arcs. A node $\mathcal{O}$ consists of different sub-objects or primitives. Each node has different attributes a, with weights w, and tolerance $T(a)$ defined as:

$$T(a) = \begin{cases} 1, & \mid a^{\mathrm{mod}} - a^{img} \mid \leq c \\ \dfrac{1}{\mid a^{\mathrm{mod}} - a^{img} \mid}, & otherwise' \end{cases} \tag{1}$$

where c is the tolerance, and a^{mod} denotes the value of attribute a in the model and a^{img} the value of the property a in the image.

Two nodes in relation according to $\mathcal{R}$. Each relation $<c,d>$ is decomposed into k subrelations between the same nodes, each with a weight v and a tolerance $T(r)$ defined analog to weights and tolerances of attributes. Figure 2 shows the resulting graph and the inner structure of nodes and arcs.



r_1 r_2 … r_q … r_k
v_1 v_2 … v_q … v_k
$T(r_1)$ $T(r_2)$ … $T(r_q)$ … $T(r_k)$
O_1
O_2
O_3
O_4
a_1 a_2 … a_g … a_n
w_1 w_2 … w_g … w_n
$T(a_1)$ $T(a_2)$ … $T(a_g)$ … $T(a_n)$
O_5 … O_i … O_n

Figure 2 Description language graph

The weights *w* and *v* are necessary for the model verification. Each of the geometrical, positional, and relational properties has a certain weight in order to verify the corresponding description to a given image. Since these weights are influenced by the data and therefore application dependent, they have to be fixed during the set-up procedure. The verification of image to description consists of verifying whether the number and type of features and primitives are the same. Next, attributes and relations are checked whether they match within given tolerances. The verification process is carried out by comparing all attributes of a node and its successors with the model. The confidence for a node can be computed based on the result of the comparison:

$$conf(p) = \sum_{g=1}^{n} w_g \cdot T(p_g) + \sum_{<p,q> \in R}^{m} v_{<p,q>} \cdot conf(q) \qquad (2)$$

By computing the consistency for different descriptions the one with the highest confidence value can be chosen if the confidence is above a certain threshold. The use of weights allows a two step identification; primitives with high weights are first detected and checked, next primitives with low weights are postulated on a certain position and verified.

2.2 Detection

The success of feature-based inspection techniques depends on the quality of feature detection. Expert systems have been developed and used to solve and refine feature detection. Problems, such as edge detection and region extraction, to name the most important in 2-d feature detection, belong to the mathematical class of inverse ill-posed problems [POG85]. There exists no unique and stable transformation function that can build a specific description starting from an arbitrary observation. To overcome this problem, one has to reduce the number of acceptable solutions by introducing a priori knowledge of the problem space on the solution space and considering the detection process as being decomposed into a sequence of sub-problems that are either well-posed problems or problems for which regularization methods exist [CRE93].

Feature detection algorithms are composed of detection components. Different components (like different edge detectors) have to be arranged to find an optimal solution in terms of quality and performance. To be able to use different components, a common interface has to be defined for the components and if different algorithms should be used to detect the same type of primitive a defined interface has to exist for algorithms too. An interface specification describes the function of the component or algorithm. The effects of the component's or algorithm's execution on program state are described in the definitions provided by the primitives.

2.3 Analysis

The description language provides the inspection model, generally the inspection could take place based on this model. Since performance and speed are crucial and the verifying phase is a graph isomorphism problem whose general case is known to have NP complexity [EVE79], the semantic information stored in nodes and arcs decreases complexity.

The analysis is formulated in a hierarchical graph structure. It represents the space of all possible problem solutions in the particular domain. A solution is formulated by instantiating the graph to form a unique solution. Each node in represents an element of the solution called *cell*, i.e. an image processing task like Sobel edge detection or a working step like image acquisition. The *arcs* between the nodes represent one of the following semantic relations:

- *Subdivide relation*: This is a n- ary relation representing a subdivision of a cell into its constituent set of sub-cells. For example image acquisition can be subdivided into CCD size, upper and lower threshold for transistor response, and frame transfer to name a few. Each subdivide relation has a weight, which allows an ordering of relations within the graph, subdivide relations with equal weights may be executed in parallel.

- *Optional subdivide relation*: This is a n- ary relation allowing optional subdivisions of a cell providing alternative subdivisions. For example, a segmentation can be performed by using region growing or optionally by split and merge. The relation is again provided with weights. If there are optional subdivisions that have to be performed together, this is denoted by an arc in the graph.
- *Detail relation*: This is a relation between a cell and a detail of the cell. For any cell there may be a number of different possible details. For example a cell representing edge detection could be detailed in Roberts, Sobel or Canny algorithm.

Each cell can have a set of in- and outputs, which can be connected to in- and outputs of other cells in the analysis graph, representing the data-flow. The hierarchy in the graph is kept, the output of one cell in one level may be interconnected to the input on the same level, the in- and outputs of cells in different levels represent the same values.

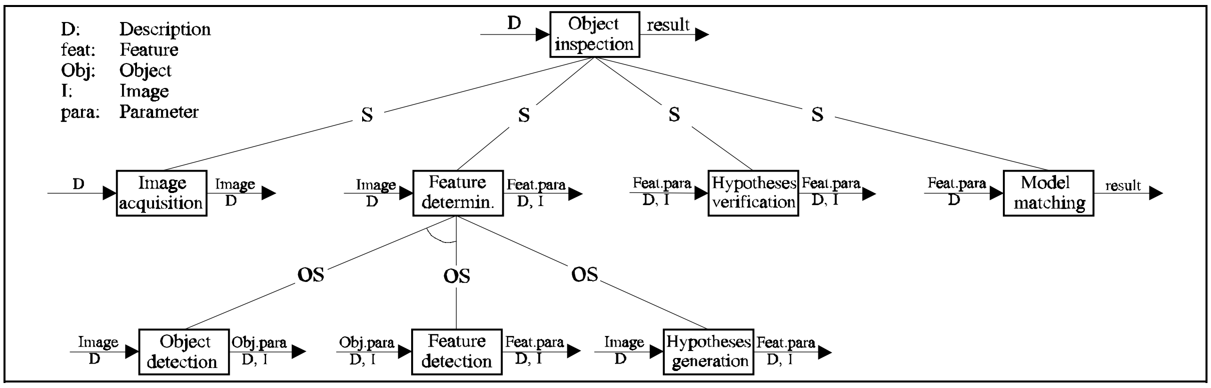


Figure 3 General analysis graph

The building up of the analysis graph for inspection can be generalized, since inspection differs only in the description and detection, the overall process is common to all of the inspection problems. Figure 3 shows the general analysis graph for inspection. The input of the *object inspection* graph is the *description*, the output is the *result* of inspection. The object inspection can be (non-optionally) subdivided into:

- *Image acquisition*: The image and its statistical parameters like noise distribution are output of this cell.
- *Feature determination*: The feature determination determines specific parameters of features (like position and size), within the image. Depending on the image acquisition, feature determination can be subdivided into:
  - *Object detection*: The object has to be located and its size has to be determined, and
  - *Feature detection*: Within the object, features used for inspection have to be located.

  Optionally, if the image acquisition parameters are fixed (the object has always approximately the same position, orientation and size) these two cells can be replaced by:
  - *Hypotheses generation*: Hypotheses about the position, orientation and size of all features are generated with the help of the description.
- *Hypotheses verification*: The specific parameters of the features, either determined by detection or by the generation, are checked whether they match with the description using features, which were not used up to this step.
- *Model matching*: The final step of analysis matches the actual parameters of the features with the description, thus producing the result of inspection.

The weights of the analysis graph are set in accordance with their probability of detection. If some primitives can be found accurately with high probability, they get a high weight. Primitives with high weights are searched first, if they can be found, primitives with lower detection probability are predicted in a specific part of the image (a hypothesis generation) and then checked for their presence (a verification of the hypothesis).

2.4 Inspection System Generation

Following the preliminary system test, the user has to check, whether the inspection system adheres to the industrial constraints. If they are not adhered to, strategies to solve the problem have to be developed. This interactive adaption of the inspection process results in a new instantiation of the analysis graph and possibly in a reduction of features to be looked for. The new inspection process has to be tested again if it attains the constraints within the testset of defect-free ("gold") and defect-images. This test determines the rate of false negatives, i.e. the number of objects classified as defective in the "gold" series, and the rate of false positives, i.e. the number of objects classified as defect-free in the defect-images series.

The final inspection system test shows if the inspection system performs correctly with defect and defect-free images, determining computation time, performance and accuracy. If one or more of the results are not satisfactory, an adaption of the inspection process has to take place. If the results are far from being satisfactory, a complete re-design of the image acquisition and illumination is necessary. If all parameters lie within the constraints, the final inspection system is ready, description language and analysis graph together with the detection algorithms are used to perform the inspection. Nevertheless, since the inspection system was set up using a limited number of images, it has to be monitored during the first operational use.

3 Calibration of Analog Display Instruments

This chapter shows the applicability of the inspection concept proposed in the previous chapter on the case study of Analog Display Instruments (ADI). This type of instrument serves as a demonstration since there are various different types of measuring instruments with innumerably different displays and layouts, but all of them have certain common properties which can be used to build up a specific description.

3.1 Primitives of Analog Display Instruments

The generation starts with a definition of the primitives of ADI’s, which are not necessarily the same as the primitives for generic detection, since non parametric primitives are added in this definition step. Three primitives describe analog measuring instruments (see Figure 4):

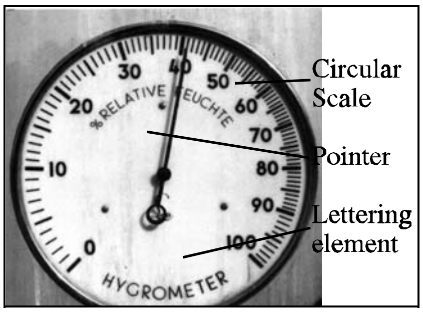


Figure 4 Primitives of a hygrometer

- *Pointer*: A pointer can have any symmetric shape such as line, triangle, rectangle or a combination of them. In addition, pointers that rotate have a circle at their center of rotation (see Figure 4). The shape is defined by a primitive, a combination of them, or in the case of a shape that is not easily represented by primitives, by a bitmap, containing one half of the shape and the medial axis.

- *Scale*: The shape of a **scale** depends on the motion of the pointer, scales with rotating pointers have the shape of a **circle** or a **circular arc**. Pointers moving straight have rectangular scales.
- *Lettering element*: They carry information about the measurement and the orientation. and includes all writings such as unit, company name, firm's symbol, maker's emblem.

The nodes of the description contain the shape features of the object. They have the following generic parameters, all of them are defined in an object-centered coordinate system referred by the origin in image coordinates:

- **Measuring instrument:** *type* (for every measuring instrument type there must be a description that allows a distinction of the measuring instrument), *shape* (shape of the instrument: circle, rectangle, triangle, free form ...), *origin* (origin of the object centered coordinate system in x,y coordinates), *size* (size of the instrument in pixel and millimeters), number of measuring units (n), number of lettering elements (m), absolute measurement value (mv_n, measurement value in measurement unit).
- **Measuring unit:** normalized measurement (c, states the relative measurements in units), measurement digits (e, states which digit of the unit is displayed), unit (u, unit of measurement of the scale), offset (o, origin of measurement).
- **Scale**: symmetric scales can have three different shapes with following parameters: *type* (circular, elliptic, or rectangular), size (radius(r), or w_e,h_e, or height(h)), origin (p_{s0}), orientation (α_{s0}), graduation (α_{sg} or d_{sg}), range (α_{sr}, or width (w)).
- **Pointer:** *type* (circular or rectangular), *shape* (bitmap or line with length and offset or rectangle with length and with), *origin*, position (α_p or p_p, variable).
- **Lettering:** *type* (bitmap, string, geom. primitive), origin (p_l, in x,y coordinates), *content* (bitmap, string, geometric primitive), *size* (width and height in pixels).

Following the definition of primitives, relations between them complete the description. The basis for the generic parameters of the spatial relations are again the results gained from generic detection. All other relations, like measurement value, have to be added. For analog display instruments following relations (conditions, equations,..) among primitives which are defined in the grammar of the description language, further rules can be added [SAB96]:

- **At the level of a measuring unit:**
  - R1: type(pointer) = type(scale)
  - R2: For circular scales only: origin(pointer) = origin (scale)
  - R3: (unit independency) $c = \dfrac{\alpha_p - \alpha_{s0}}{\alpha_{sg}}$ or $c = \dfrac{p_p - p_{s0}}{d_{sg}}$
  - R4: (range constraint) $|\alpha_p - \alpha_{s0}| \leq \dfrac{\alpha_{sr}}{2}$ or $|p_p - p_{s0}| \leq \dfrac{w}{2}$
- **At the level of a measuring instrument:**
  - R5: Measuring units on the same measuring instrument are related through their position. Note that different measuring units may be overlaid at the same position.
  - R6: Lettering elements are related through their position
  - R7: (measurement digits) $e_i = c_i * u_i + o_i$ where $i = 0..n$ denotes the measuring unit
  - R8: (measurement) $m_v = \sum_{i=0}^{n} \dfrac{\lfloor e_i \rfloor}{\prod_{j=0}^{i} u_j}$ where $u_0 = 1$

Following the definition of primitives, relations, and generic parameters, the graph forming the description language is defined. The hierarchical structure is built up in a top-down manner; it starts with the abstract class of measuring instrument and ends up in the primitives. Figure 5 shows the description for ADI's including parameters.

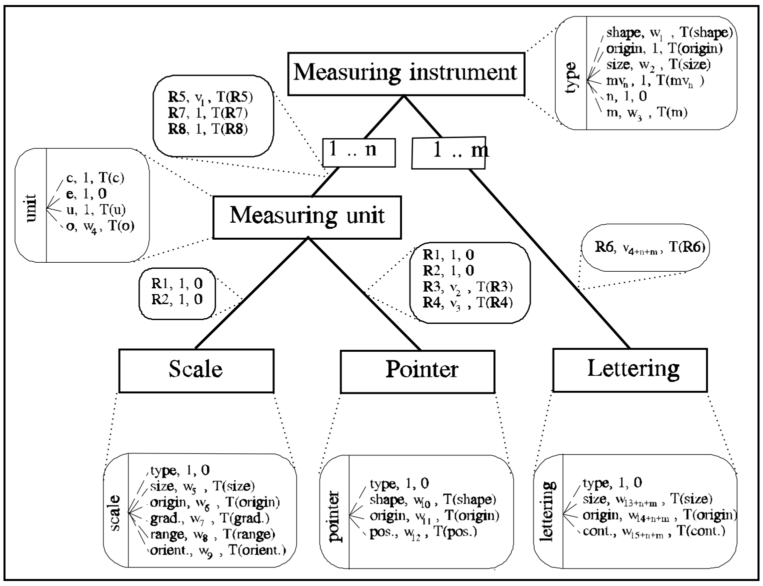


Figure 5 Description language for analog display instruments

The root of the graph is the primitive "Measuring instrument", which is subdivided into *n* "Measuring units" and *m* "Lettering" elements. A common instrument like a hygrometer has only one measuring unit, but a clock for instance has normally 3 measuring units. Therefore, this sub-hierarchy is necessary. Every measuring unit is subdivided into one "Pointer" and one "Scale", all the previously defined primitives are the leaves of the graph. This general description language allows a representation of any analog display instrument in a separate graph, constructed out of the results of the generic detection. Weights and tolerances are added during the analysis step.

Figure 6 shows an example for a manometer with a temperature scale. The description results in two different measuring units, temperature and pressure.

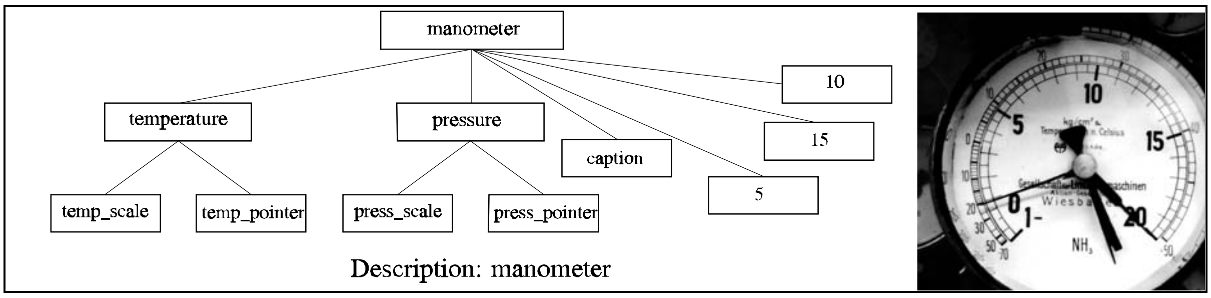


Figure 6 Examples for specific descriptions

3.2 Analysis Graph for Analog Display Instruments

The next step in generating an inspection system for ADI's is the construction of the analysis graph. The general analysis graph is applied to the specific problem. In- and output parameters are described at the top level of the graph only, to simplify the graph. This general concept is used for the definition of the analysis graph for ADI's.

For the example of manometer inspection, the imaging parameters are supposed to be unknown. Therefore, the analysis graph of the manometer (Figure 7) has two subdivisions of the node *feature determination*. First the manometer is detected using a circle detection, defining the origin of the object-centered coordinate system. Since the origin of the two scales is the same as the origin of the instrument, only one lettering element has to be looked for to determine the orientation of the scales (i.e. "NH_3" is looked for using a pattern matching technique along a circle in a known distance). To verify the result, the lettering element "15" is checked by *hypotheses verification*, i.e. using an OCR algorithm. If the test is successful the positions of the 2 pointers are determined. In this example pointer 2 cannot be detected since it is outside of the scale range. Since the orientation of the scales and the position of one pointer are known, the measurement unit determination, computing the angle between the origin of scale 1 and the pointer 1 is performed. In this example, the measuring instrument value determination has the result mv_1= -0.01 and mv_2= not valid.

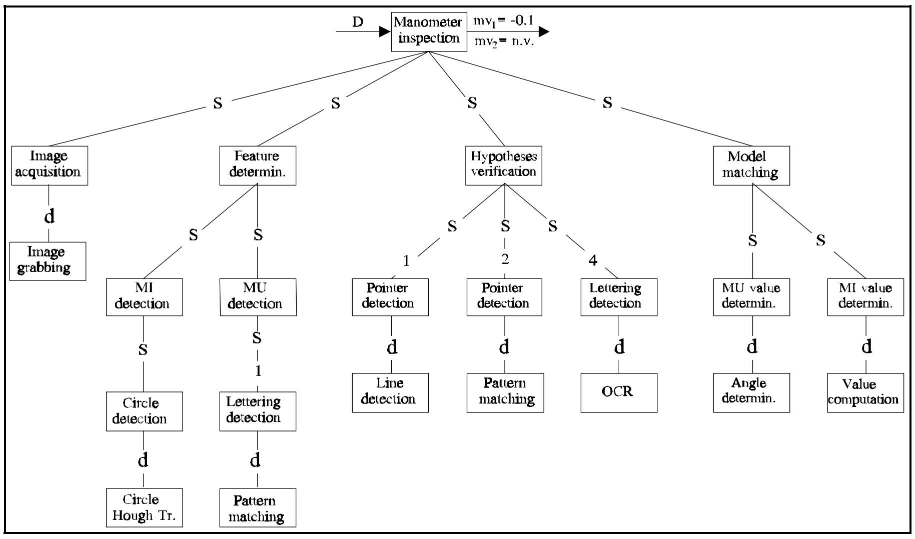


Figure 7 Analysis graph: manometer

3.3 Results

Two examples were chosen to show that the description language and the analysis graph together with detection algorithms can handle different types of analog display instruments:

1. *Hygrometer*: The appropriate description and analysis graph were used. Furthermore, two circular lettering elements to define the orientation and the pointer are detected. Figure 8 shows the result for the test image; a humidity of 41% was the correct result.
2. *Watch*: With the description of the clock, generated by interactive definition of the primitives the analysis produced the result shown in Figure 9. In our test series (20 samples) all pointer positions were computed exactly, the time was correctly read in all images.

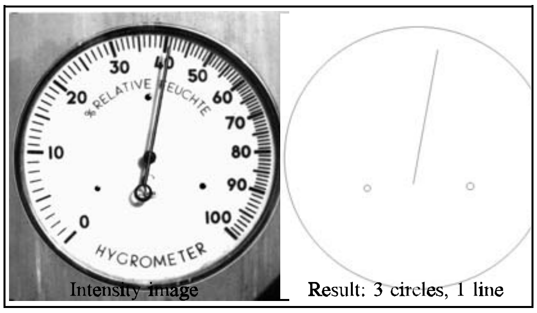


Figure 8 Result hygrometer (mv_n= 41% hum.)

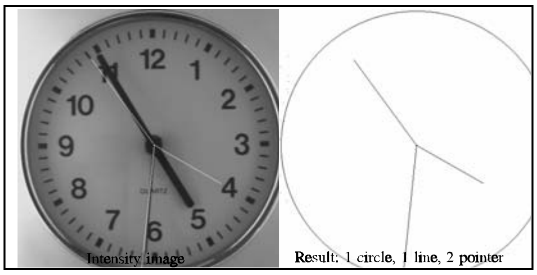


Figure 9 Result for a watch (mv_n = 4:54:31 h)

The working time for adapting the description and the analysis graph interactively was approx. 1.5 hours each for the hygrometer and the clock. To solve the problem of overlapping pointers, the minutes hand was searched first. If the other hands could not be detected, they were assumed to be overlaid by the minutes hand.

The examples were simulated applications, therefore concrete data about reliability and accuracy cannot be given. Image acquisition, illumination, and image series with golden and defect images of different objects of the same type should be used to determine these parameters. However, the three additional examples showed that the claimed flexibility of the concept is given by adapting description and analysis graph to the specific object.

4 Conclusion

In this paper a **concept for visual inspection** has been presented in which the **detection** of primitives was separated from the **analysis**. This separation was achieved by defining a general **analysis graph** for inspection, containing detail relations that represent detection algorithms. Existing pattern recognition software was re-used for detection. The use of any detection algorithm was possible by changing the analysis graph instantiation in the detail relation, the overall analysis process stayed the same. Together with an object specific description, defined in a so called **description language**, the analysis graph was instantiated. This systematic approach to inspection allows its application to a wide range of inspection problems. It can be seen as a "recipe" for solving industrial applications, stating at which stage which kind of decisions have to be made. The systematic approach also permits a high degree of flexibility since it contains application specific and application independent parts.

The applicability of the inspection concept was demonstrated on the case study of analog display instruments. This type of object served as a demonstration since there are various different types of instruments with innumerable different layouts, but all of them have common properties which were used to build up a specific description language. Examples demonstrated that the ADI analysis graph can be used to inspect specific instruments (hygrometer and clock).

5 References

[AME95] Amerinex Artificial Intelligence Inc., 409 Main Street, Amherst, MA 01002, USA, "*The KBVision System - User's Guide*", 1995.

[BAT96] B.G. Batchelor, P.F. Whelan, "Machine vision systems: Proverbs, Principles, Prejudices and Priorities", *Proc. of SME Conf. on Applied Machine Vision*, Cincinnati, pp. 7-19, 1996.

[BOD95] R. Bodington, "A Software Environment for the Automatic Configuration of Inspection Systems", *Proc. of Intl. Workshop on Knowledge- Based Systems for the (Re)Use of Program Libraries*, Sophia Antipolis, France, Vol.1, pp.100-108, 1995.

[BUN92] H. Bunke, A. Sanfeliu, "Statistical and Syntactic Models and Pattern Recognition Techniques", in: C. Torras (Ed.), "*Computer Vision: Theory and Industrial Applications"*, Springer Verlag, New York, pp.215-266, 1992.

[CHI92] R.T. Chin, "Automated visual inspection algorithms", in: C. Torras (Ed.), "*Computer Vision: Theory and Industrial Applications"*, Springer Verlag, New York, pp.377-404, 1992.

[CRE93] D. Crevier, "Expert Systems as Design Aids for Artificial Vision Systems: A Survey", *Proc. of SPIE*, Vol.2055, pp.84-96, 1993.

[DAR88] A.M. Darwish, A.K. Jain, "A rule-based approach for visual pattern inspection", *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol.10, No.1, pp.56-58, 1988.

[EVE79] S. Even, "*Graph Algorithms*", Computer Science Press, Rockville, 1979.

[FRE89] H. Freeman, M.Y. Chiu, D.D. Dreyfuss, I. Gorog, "Is Industry Ready for Machine Vision?", in: H. Freeman (Ed.), "*Machine Vision for Inspection and Measurement*", Academic Press, Inc., 1989.

[FU82] K.S. Fu, "*Syntactic Pattern Recognition and Applications*", Prentice- Hall, Englewood Cliffs, New Jersey, 1982.

[KHO95] Khoral Research Inc., USA, "*KHOROS Program Services - User's Guide*", 1995.

[MAR92] A.D. Marshall, R.R. Martin, "*Computer Vision, Models and Inspection*", World Scientific, 1992.

[MAT96] Matrox Electronic Systems Ltd., 1025 St. Regis Blvd., Dorval, Quebec, H4P 2T4, Canada, "*Matrox Imaging Library (MIL) - Product Overview*", 1996.

[NEW95] T.S. Newman, A.K. Jain, "A Survey of Automated Visual Inspection", *Comput. Vision, Graphics, Image Processing*, Vol.61, No.2, pp 231-262, 1995.

[POG85] T. Poggio, C. Koch, V. Torre, "Computational Vision and Regularization Theory", *Nature*, Vol. 317, pp. 314-319, 1985.

[SAB95] R. Sablatnig, "A highly adaptable Visual Inspection Concept by Separating Detection and Analysis Process", *Proc. of Intl. Workshop on Knowledge- Based Systems for the (Re)Use of Program Libraries*, Sophia Antipolis, France, Vol.2, pp.25-27, 1995.

[SAB96] R. Sablatnig, "Flexible Automatic Visual Inspection based on the Separation of Detection and Analysis", *Proc. of 13th International Conference on Pattern Recognition, Vienna*, Vol. 3, pp. 944-948, IEEE-Computer Society Press, 1996.

[TOD88] J.D. Todd, "Advanced Vision Systems for Computer-Integrated Manufacture -Part 1", *Computer Integrated Manufacturing Systems*, Vol.1, No.3, pp.143-154, 1988.