User Programmable Visual Inspection

J. Jeffrey Hunter, Jim Graham and Chris J. Taylor
Wolfson Image Analysis Unit
University of Manchester, UK
jeff@wiau.mb.man.ac.uk

Abstract

The usefulness of advanced machine vision techniques for automated inspection is restricted by long development cycles for the inspection software and a lack of general applicability of the resulting system. These difficulties arise because in the majority of cases the particulars of the inspection are embedded in the software structure resulting in severe restrictions on user reconfiguration. This paper describes the structure and major components of an inspection system which is capable of tackling a wide range of inspection problems without reprogramming. User configurability is based around a simple, interpreted geometric description language which provides support for both dimensional and surface quality measurements. The operation of the system is illustrated using the inspection of automotive brake assemblies as an example.

1 Introduction

There is considerable potential for the use of automated visual inspection in manufacturing industry. Some inspection problems require specifically tailored algorithms to achieve satisfactory results [1] but there is a large class of problems which can be simply stated in terms of object location, dimensional measurement and surface quality assessment. Members of this class can be loosely termed as "generic inspection problems".

In the past, practical inspection systems have typically relied on embedding knowledge of the target inspection task in the algorithms [2]. This approach tends to limit the reusability of the system and certainly ensures that there is no possibility of the system or its subparts being reused by anybody other than a vision expert. This means that new inspection systems are expensive to develop often precluding on cost grounds applications where, otherwise, visual inspection would be an ideal solution. Most inspection system developers appreciate these difficulties and attempt to surmount them by creating inspection libraries. Libraries go some way to providing for simpler development but they are generally low-level entities still only of value to the vision expert. What is required is a system in which the user's description of the task is separated from the image analysis. We have achieved this and constructed an inspection system which is flexible and user configurable giving the user the ability to tackle a wide range of generic inspection problems.

BMVC 1994 doi:10.5244/C.8.65

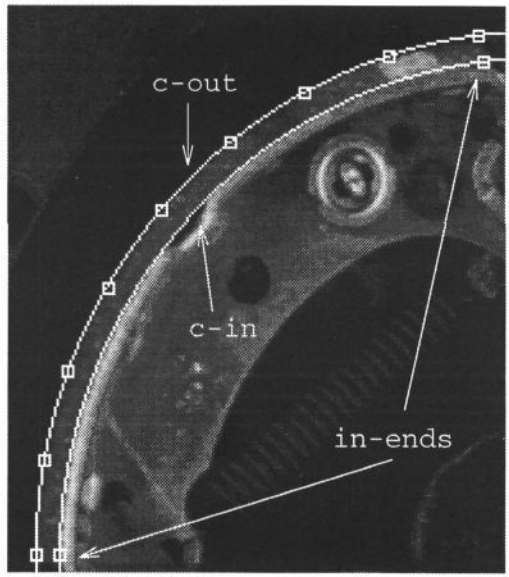


Figure 1: A display obtained from the system during inspection prototyping (with manually added annotation)

We describe the system's components and operation and show how a user-specified task can launch complex image analysis operations. We demonstrate how our system allows generic inspection tasks to be to be rapidly prototyped by a domain expert and can be used to perform new inspections tasks with minimal effort.

To assist the reader in understanding the material presented we have illustrated the steps in solving a typical inspection problem using our system. Figure 1 shows part of a brake assembly which is the object to be inspected. One of the inspections is a test for a departure from circularity of the lining on the shoe. To perform this inspection the user would initially provide the system with some example training images with the objects identified by landmark points. Then, using the task description language, he/she would describe the interpretation task. In this case language statements would be used to tell the system that a circle should be fitted to the lining boundary landmark points and that a calculation of the distance between these boundary points and the circle is to be made. Once the description is complete the user can execute the inspection by requesting the value of the distance.

2 The Inspection System

Visual inspection consists of a number of stages from image acquisition through to process control. In this work we are interested in that part of the inspection process which is concerned with the calculation of measurements from images of the components to be inspected. In our design this subsystem comprises of three

elements: object search, geometric interpretation and surface property evaluation. These components are discussed individually in the following subsections.

2.1 Object Search

It is clear that to create a generic inspection system we require a generic object search technique. We have used the Active Shape Model (ASM) technique described by Cootes et al [3]. They compare this approach with a number of other methods for image interpretation based on flexible or deformable models and identify the strength of the ASMs [4] as arising from their specificity to the particular object for which they are searching. Specificity gives an ASM robustness in the face of noise and clutter in the image but from our point of view the general applicability of the technique is even more important. This generality arises from three key characteristics of the approach.

Firstly the method relies on the identification of object landmark points. Landmark points are positions which can be unambiguously identified in all examples of the object. Most objects which are the subject of inspection have identifiable landmark points because these are the points which define the object's dimensions. Secondly the method records statistics of the relationships between the positions of the landmark points in a set of training examples. The statistics which are recorded make no assumptions about the form of the object. Finally, and a major reason why the ASM technique has a fundamental role to play in the creation of a generic inspection system, the technique treats the spatial relationships between multiple objects in an assembly naturally. Many inspection tasks consist of both inter and intra object measurements and the ASM is able to deal with both in a uniform and configuration-independent way.

A part of every inspection task is determining whether the assembly has components missing. In this respect a further useful capability of an ASM is its ability to provide a measure of confidence that it has located an instance of the target object in an image. These measures can be turned into boolean existence variables by the search module and made available on a component by component basis to the rest of the system.

2.2 Geometrical Construction

There are two common types of measurement required in generic inspection: measurement of the dimensional properties of components and measurement of the material quality or finish. The dimensional properties are generally compared to design tolerances. The material quality is assessed over critical regions of the assembly.

Our inspection description is based around geometries because tolerances are generally expressed in terms of the dimensions of the geometries used in the design whilst areas of the assembly which require surface assessment can also be described by geometric constructions.

Figure 2(a) provides an idealised example to demonstrate the points to be made in the following sections. The figure shows an assembly which is a sheet with a hole cut from it. Consider the task of measuring the angle of the top left

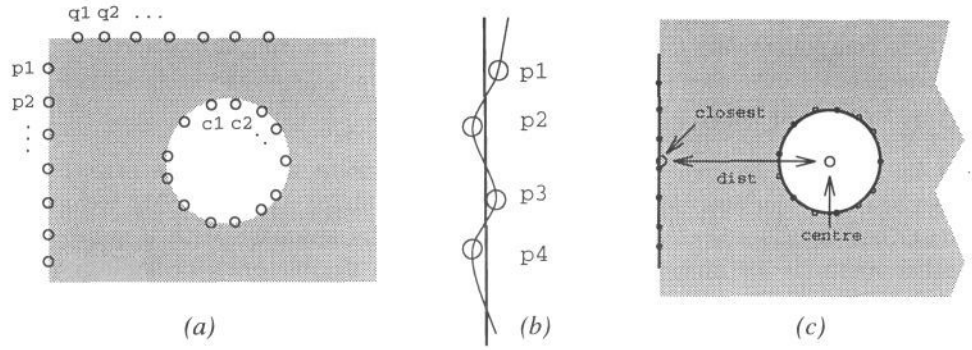


Figure 2: (*a*) An example shape for inspection (*b*) A close up of the landmark points on the edge. (*c*) An example measurement using the `closest` and `distance` operations.

corner. When an inspection is performed on the object the image search technique is used to locate the landmark points on the sheet's rectangular boundary and the edge of the circular hole. A close up view of the object edge (2(b)) shows that it has undulations and that the object search has performed accurately. The angle measurement depends on the considering the two edges as straight lines as in the original design. The inspection description therefore reconstructs a straight line from the points which have been found.

In general the geometric task description consists of describing the construction of geometries from other geometries using operations. In the example above we have constructed lines `l1` and `l2` from the edge points.

```
l1 : line([p1,p2,p3,p4])
l2 : line([q1,q2,q3,q4])
```

The angle can then be calculated from the lines

```
a : angle(l1,l2)
```

In the following sections we discuss what types are required to perform a range of inspections and what operations should be available to act on each type. We also show how descriptions can be written to handle intentional variations in assemblies and how task description are executed to obtain results.

2.2.1 Types

Only a small range of geometries and other types are required within the system to support a wide set of inspection tasks. The current system includes points, lines, circles, regions, booleans, reals and groups. It is accepted that this list may not be complete but the important point is that only a few more geometric types are required to tackle the majority of cases. A previous study [5] of 17

industrial production-line inspection problems covering a wide range of products identified the requirement for only three extra geometric types: boundary, ellipse and hyperbola.

Of the types in the system, groups, regions and booleans deserve individual explanation. Groups provide a way of collecting together sets of possibly mixed type values and treating them as a single entity. A region is a geometry defined by a group of boundary points and may represent an irregular connected area of the image. Boolean types are used in the logical control of *if then else* constructs in the description.

2.2.2 Operations

It is object search which locates the landmark points on objects in the image. Operations provide the mechanism for constructing new geometries based on these points. Operations are the basis for the user description of the whole inspection task.

We have already seen (figure 2(b)) operations used to describe the construction of a line from a group of points and the subsequent calculation of the corner angle. The operations provided by the system are for the most part restricted to those which act on or produce geometric types; the number required to provide satisfactory functionality is limited. It is expected that extra functionality such as arithmetic capability will be provided by links to other more appropriate tools.

There are 17 user operations provided in the current system: **line**, **point**, **circle**, **region**, **[]**, **if**, **closest**, **distance**, **derive**, **min**, **max**, **sd**, **mean**, **less**, **greater**, **between** and **greybox**. From this list of functions two of the operations which have been identified as providing real descriptive power are **closest** and **distance**. These are both overloaded functions and can take the following forms.

`closest(<any geometry>,point)` returns the position on the geometry which is closest to the point.

`closest(<any geometry>,[points])` returns a group of positions, one for each point in the group `[points]`.

`closest(<any geometry>,<any geometry>)` returns the point on the first geometry which is closest to the second geometry

Similar definitions hold for the distance operation

```
distance(point,point)    distance(point,[group])    distance([group],[group])
```

As an example of the operation of distance and closest we will extend the example of figure 2 to measure the distance between the centre of the cut out hole and the left edge of the assembly (see figure 2(c))

```
hole        : circle([c1,c2, ...])
centre      : hole.centre
closest     : closest(edge,centre)
dist        : distance(closest,centre)
```

2.2.3 Execution

The geometrical descriptions in the inspection system can be thought of as forming an acyclic dependency graph. Edges in the graph correspond to operations and vertices to the values in the task. Evaluation is lazy and prompted by a request for the value of some entity in the task. In the simple example above it might be a request for the value of `dist`. Execution recursively evaluates all the dependencies until it reaches the landmark points. The system then makes a request to the object search module to evaluate the position of the required landmark points. The object search selects the models relevant to the required points and initiates the search. When the search is complete the search module returns either the point locations or an indication that the points cannot be found. After the landmark points have been determined the recursive dependency calculation unwinds. The values of each of the dependent nodes are calculated until finally the requested value can be returned.

To provide the user with a fast prototyping ability the system is interpretive; changes to inspection task description are effected immediately. Out-of-date management is built into the execution strategy to avoid unnecessary computational effort when changes occur.

2.2.4 Variants

There are situations in inspection where the components to be inspected may not all be identical but may be any one of a set of variants. In our example of brake inspection the production line contains a mixture of right and left hand assemblies. There is no prior information as to which variant is currently being inspected.

The notion of existence has already been discussed in object search. The existence of certain objects can be used to determine which variant is being inspected. Returning to the example of figure 2 we could assume that a variant condition was that an elliptical hole was cut in the sheet instead of a circular one. Thus we might define

```
c-hole : circle([c1,c2, ...])
e-hole : ellipse([e1,e2, ....])
```

We require a method for dealing with variants without having to write a separate task description for each possible case. In the example above, once we have fitted the appropriate geometry to the hole the rest of the task is identical.

The `if` construct provides a facility for the selection of one of two objects on the basis of a binary value.

```
if ( <binary value>,<object (true)>,<object (false)>)
```

We can use the if construct in our example to select the appropriate object for the rest of the task to work on.

```
hole   : if (exists(c-hole),c-hole,e-hole)
centre : hole.centre
```

When this is executed the point **centre** will receive the position of the centre of whichever type of hole is found in the currently inspected sheet.

2.3 Surface Property Measurement

In the introduction we identified two major aspects to generic industrial inspection. The first was dimensional measurement and the second was surface quality measurement. As with dimensional measurement the objective in surface property measurement is to allow flexibility, adopting a methodology which is user-programmable and user-understandable. We achieve thus using the greybox operation.

The greybox surface measurement operation (as opposed to blackbox where the user has no control over the measured properties) is specified like any other operation in the inspection task.

```
greybox(<geometry>,[<properties>])
```

The property list contains a list of properties which the user thinks are relevant to the inspection of the area. The words are in a language familiar to the user, eg roughness, streak, blemish etc. For example the user may define

```
surface-vals : greybox(pad-region,[blemish])
```

Internally the words are associated with particular texture operators, eg Laws texture energy features [6].

Every geometric type defined in the system is expected to be able to supply a set of points, at a specified resolution, describing the area in the image which the geometry covers. This set of points is used as the region of application for the selected texture operators. The greybox operator returns a group of numbers which represent the outputs of the texture operators supplied in the list. We desire as much autonomy as possible from the greybox operation. There is therefore no definition of the contents of the group; all that is known is that it is a set of measures related to the properties specified. The group of numbers are fed directly to a classifier for analysis.

2.4 Classifiers

We need classifiers not only for surface measurement but also for generating decisions from a geometrical measurement. Often these decisions will be determined by allowed tolerances, but when such tolerances are not available results may be specified by training using examples of both pass and fail classes. For example, we may wish to classify on the basis of the angle between the edges in figure 2.

```
pass : classify(angle-size,a)
```

where **a** is the observable used to produce a boolean result named **pass**. Similarly for inspection of surface quality we have

```
pass : classify(pad-surface,surface-vals)
```

```
# p_in and p_out are the points
# on the lining edge

c_out : circle(p_out);
c_in : circle(p_in);

# calculate the distances from each
# of the edge points to the circles

out_best : closest(c_out,p_out);
out_dist : distance(p_out,out_best);

in_best : closest(c_in,p_in);
in_dist : distance(p_in,in_best);

# get the mean value of the distances

out_dev : mean(out_dist);

in_dev : mean(in_dist);

# calculate the points on the inside
# circle closest to the outside points

in_ends : closest(c_in,
[out_best.0, out_best.8]);

# calculate the difference between the
# separation of the circles at the two
# ends of the lining.

end_dist : distance(in_ends,out_ends);
ends_dev : sd(end_dist);

# summarise the results.

results : [out_dev,in_dev,ends_dev];
```

Figure 3: The key statements required to measure circularity and co-circularity of the brake shoe lining

When this description is executed the `pad-surface` classifier will classify on the basis of the surface properties held the `surface-vals` observable. In this case `pass` will become a boolean value indicating the acceptability of the surface.

3 Example Applications

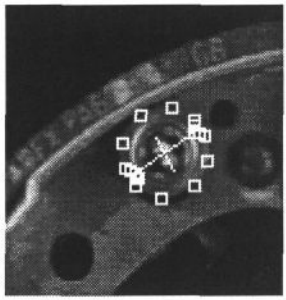
We have used the inspection of brake assemblies as an example of a complex inspection task for which we have direct experience of a hand-crafted solution [2]. This exemplar highlights many of the common requirements of an inspection: checking to see if components are present, measuring the positions of components and determining whether surfaces are finished properly. We have selected three subtasks to demonstrate different aspects of the problem.

Brake Lining Thickness

One of the requirements of the inspection task is a measurement of the thickness, circularity and co-circularity of the brake shoe surfaces. Figure 3 shows the geometric task which was entered at the command line to perform the operation. At any time during the prototyping of the inspection the system is able to display the geometric constructions and print task values. Figure 1 was obtained by entering the following statement at the command line.

```
Display c_inner, c_outer, best_outer, inner_ends;
```

The system updates any values automatically before displaying or printing them. If a new image is loaded and the statement `Print results` is issued the system will search the image for the new positions of the objects and recalculate all the required values before returning with a result.



```
# define the centre line of the pinhead and the
slot
# cap and pinhead are aliases for the associated
# groups of boundary landmark points.

cap_l :  line(cap.0,cap.8);
pinhead_l :  line(pinhead.3,pinhead.9);

# measure the angle

angle :  angle(cap_l,pinhead_l);
```

Figure 4: Determination of the location of the pin head in a blind slot. The image shows the two lines and the cap landmark points as found by the object search.

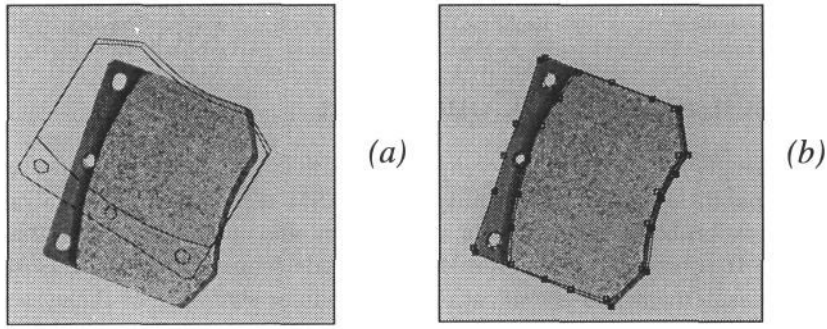


Figure 5: (*a*) The position of the model before search begins. (*b*) The final result.

Retaining Pin Location

The second example from the the exemplar is a check to see whether the shoe retaining pin is positioned correctly in the cap. The description and an example of the display are shown in figure 4. Geometric descriptions tend to be compact and apart from training the model the text shows all the statements which are are required to perform the angle measurement. If either the pinhead or cap cannot be found in the image then the system returns the angle as undefined.

Brake Pad Oil Staining

To give an example of surface inspection we demonstrate the construction of a new task on a similar theme, using disk brake pads. Figure 5 shows the object, a front brake pad from a caliper brake. A number of these pads were prepared with oil stains and the following short task was created to inspect the objects. The task statements with comments are shown in figure 6. Notice that the the rotation attribute in the model definition line tells the system that it should search for the assembly at any angle.

Setting up this new inspection, including preparation of the training examples, took only a few hours and most importantly involved no addition or change to the inspection system software whatsoever.

```
# define the parts which are to be
# searched for in the image.

Models pads [[rotate=on]("Outside",
"Inside", "Hole 1","Hole 2","Hole 3")];

# define a classifier to use to
# classify the surface.

Classifiers (oil);

# define the lining to be a region
# covering the area of interest
# ie.  the area bounded by the
# "Inside" landmark points.

lining :  region(pads."Inside");

# measure the surface quaility using
# a grey box operation

values :  greybox(lining,[blotch]);

# finally classification is performed
# leaving oil_stained with the value
# true if the pad surface is stained.

oil_stained :  classify(oil, values)
```

Figure 6: The task required to determine if the brake pads shown in figure 5 are oil stained.

4 Discussion and Conclusion

Not surprisingly generic tools are composed of individual parts which have strongly generic capabilities. The applicability of the ASM technique to a wide range of image search problems is critical in this respect and underpins the system performance. If subsequent improvements are made to individual aspects such as texture measurement and classification, the modular structure of the system will allow them to be easily incorporated.

In summary by employing suitable generic components and a modular construction we have been able to construct a visual inspection system which is user configurable but capable of tackling a wide range of inspection problems. Our main objective was to demonstrate that a generic user programmable inspection system is feasible. We believe we have achieved this, demonstrating the functionality of the system in complex inspection tasks.

References

[1] N. Bryson, R. N. Dixon, J. J. Hunter, and C. J. Taylor, "Contextual classification of cracks," in *British Machine Vision Conference 1993* (J. Illingworth, ed.), pp. 409–418, BMVA Press, 1993.

[2] P. W. Woods, C. J. Taylor, D. H. Cooper, and R. D. Dixon, "The use of geometric and grey-level models for industrial inspection," *Pattern Recognition Letters*, vol. 5, pp. 11–17, 1987.

[3] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models - their training and application." CVGIP: Image Understanding, in press.

[4] T. F. Cootes, C. J. Taylor, and A. Lantis, "Active shape models: Evaluation of a multi-resolution approach to improving image search." BMVC 94.

[5] C. B. Jackson, "Application generator development." Automated Visual Systems Ltd, University of Manchester, internal report.

[6] K. I. Laws, "Rapid texture identification," *SPIE Image Processing for Missile Guidance*, vol. 238, p. 376, 1980.