Toward Generic Model-Based Object Recognition by Knowledge Acquisition and Machine Learning

Julian Kerr and **Paul Compton**
University of New South Wales
Sydney NSW 2052, Australia
{juliank, compton}@cse.unsw.edu.au

Abstract

The aim of this paper is to present a system that uses both inducted hypotheses and expert knowledge for recognition of objects within digital images. Classification of pixels by spectral and spatial features is learned via example. Recognition of objects within resulting class images is performed by a novel method of search that takes advice on object properties, and enables error in object recognition to be systematically isolated and rectified. Guided by the system, a user selects whether error correction is machine-learned or user-defined. The adaptability of mixed-initiative error correction leads to wide applicability. The system is demonstrated using images from a robot soccer domain.

1 Introduction

Artificial systems for image interpretation are successfully employed in a number of domains such as human face recognition, control of robots for the purpose of playing soccer, and identification of cancer in breast x-rays. Solutions are typically constructed for a given task, and are unlikely to be applied to others. Such *purposive* or *goal-directed* vision systems [Aloimonos, 1990] are fundamentally brittle, and cannot deal with changes in the nature of input images.

On the other end of the computer vision spectrum is the reconstructionist vision paradigm [Marr, 1982], which speaks of a "general" vision system with capabilities such as our own. While effective biological systems obviously exist, an artificial system of comparable merit has yet to emerge.

Lying between these two extremes are a class of *generic* systems. These can undertake a variety of different tasks, or may be reconfigured to do so.

The proposed system, SanityCheck[1], is generic via adaptability achieved through learning. Machine learning is used at the pixel level for region-based image segmentation. Knowledge acquisition is used to capture object-level domain knowledge that may be difficult to reliably induct, and has traditionally been embedded during system development.

[1]The term "sanity checks" is used by the University of New South Wales robot soccer team to describe rules used for recognising objects.

Combining learning at multiple levels levels can make error correction more challenging as it is not always clear which level is at fault. Incremental Exception Learner (IEL) [Kerr and Compton, 2002] may be employed to assist in this task. In the SanityCheck setting, IEL can help a user to gauge whether a correction to the pixel classifier should be inducted, or whether an amendment should be made at the object level of the system's knowledge base.

2 How Generic is Generic?

Visual recognition of objects involves cognitive analysis at a number of levels. In identifying the type of fruit known as *apple*, we take into consideration colour, shape and contextual information (surrounding objects). We would likely think twice before biting into a metallic, apple-shaped object, or an apple-coloured cube. Alternatively, we would not gain much sustenance from what appears to be a perfect, blemish-free specimen in a bowl on a table inside a furniture shop.

SanityCheck takes into consideration knowledge at three levels. The feature space employed at each level, and hence the scope of applicability of this generic system, appear in 2.1, 2.2 and 2.3. Details of learning processes follow thereafter.

2.1 Pixel Level Feature Space

Artificial analysis at the pixel level is based on descriptions of the spectral and textural properties of an image.

Spectral properties relate to individual pixel values. For grey-scale images, this equates to pixel intensity. In colour images, the spectral properties of a pixel are its values for each dimension in colour space. For some remote-sensing applications, each pixel may be described by data from a variety of spectra.

Texture is a descriptor that relates to the statistical, structural and spectral properties of a group of adjacent pixels [Gonzales and Woods, 1993]. The system uses first-order statistics (information about intensity distribution) and second-order statistics (spatial information of the spectral distribution) with which to model texture.

For each pixel of an input image, the system outputs a class value based on its spectral and textural properties. The result is a *class map* of the same dimension as the input image. Refer to figure 1.

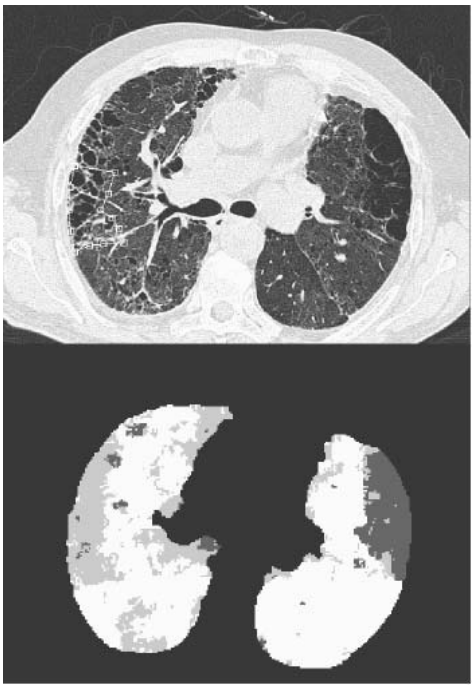

Figure 1: Input Image and Class Map of Lung Regions

2.2 Object Level Feature Space

Once a class map has been produced, the system connects adjacent pixels of like class into "blobs". The underlying assumption of this strategy is that objects can be described in terms of sets of homogeneous regions.

The blob is the base unit of this level of analysis: an object is modelled as consisting of one or more component blobs. Each component is defined by a set of required blob properties and relationships with other blobs.

Examples of blob properties:

- Class
- Area, length, width, aspect ratio
- Direction of principle axis
- Level of curvature
- Shape descriptors[2]

Examples of blob relationships:

- Distance
- Ratio of blob properties (such as area ratio)
- Relative location

Refer to the goal and ball shown in figure 2. Each object consists of a single component. The ball is an orange blob with an aspect ratio of 1. The goal is a blue blob with an aspect ratio of 2.

[2]The programmatic implementation of blob property tests follow a common interface. Hence property tests such as shape descriptors could be based on such techniques as an analysis of chain codes, or an artificial neural network analysis of blob pixels. Equivalently, an interface also exists for blob relationship tests.

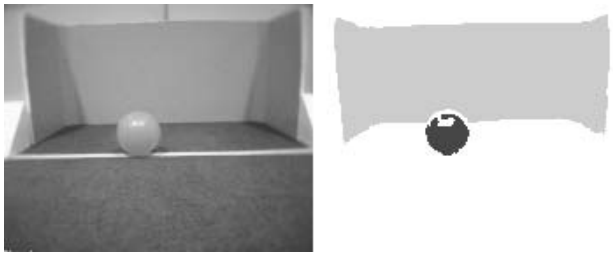

Figure 2: Goal and Ball

Figure 3: A Beacon

A beacon, as appears in figure 3, is modelled as two components: *top* and *base*. Component *top* is green, and has an aspect ratio of 1. Component *base* is pink, has an aspect ratio of 1, is located within a distance of 1 pixel of component *top*, is located below *top*, and the ratio of area with area of *top* is 1.1.

Refer to section 5 for details of creating, and incrementally updating, object definitions.

2.3 Global Level Feature Space

Knowledge at the global level consists of relationships between objects. For example, assuming the ball does not leave the ground, a rule could be added stating it is not possible for a ball to be located above a beacon. This might be useful in discriminating the real ball from an orange cap being worn by a spectator.

An object is comprised of a set of one or more component blobs, and like a blob it occupies a region within an image. Hence the set of relationships at the global level contains the same relationships as the set for blobs: namely distance, ratio of areas, relative location etc.

3 Learning at the Pixel Level

Recognition of colours and textures within digital images is performed automatically by our visual system, without conscious consideration of the properties of each pixel. We cannot relate the knowledge we employ to discriminate colour and texture, but we can teach by providing examples. Modest user interaction readily generates training data, with an instance derived from each labelled pixel.

Machine learning has been applied to pixel-level analysis in a number of domains [Kerr and Wilson, 2001], [Ojala and

Pietikainen, 1999], [Uppaluri *et al.*, 1996]. Typically, a user provides labelled regions of interest to the system, which creates training instances based on the properties of enclosed pixels. Supervised learning then provides a classifier that is used for generalising unseen image data.

SanityCheck also follows this scheme, and may use IEL as its pixel-level classifier. In doing so, the system can offer feedback to assist the user in deciding whether it is better for error correction to be generated automatically from example data, or to be defined explicitly by the user. A discussion of this mixed-initiative approach to error correction is presented in section 6.

4 Incremental Exception Learner

4.1 IEL Structure

A trained IEL takes the form of nodes connected in a tree structure of variable branching factor. IEL nodes consist of a set of instances of the node's class, C_{node}, and a binary classifier trained to distinguish between these instances and those of class C_{parent} of the node's parent. $C_{node} \neq C_{parent}$. Refer to figure 4.

An instance will cause a node to fire if the node classifier outputs class C_{node} over class C_{parent}.

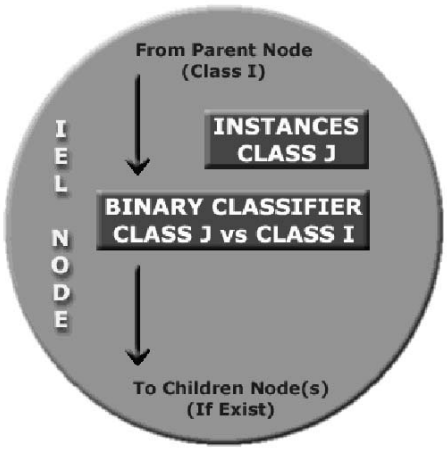

Figure 4: An IEL Node

4.2 Induction

Initial formation of an IEL tree takes place as example data from a default class are provided to the learner. A root node containing this data is generated. No classifier is associated with this node, and the condition for firing is always true. Hence the system will initially conclude that the universe consists only of the default class.

Modification of the IEL tree occurs each time the system makes significant classification error. Identification of error and its correction (providing training data) are performed by the user. A node that produces an incorrect conclusion is corrected by the addition of a child node. A child node is assigned all the instances of a class that were misclassified by its parent. The binary classifier within a child node is trained to distinguish between the instances of the child and those of its parent.

4.3 Example: Iris Plants Database

The Iris Plants Database is a simple domain consisting of 3 classes, 50 instances of each class, and 4 numerical attributes. For readability, the following example uses C4.5 as the inductor for IEL nodes. Learning is performed in three steps:

1. Iris-versicolor is selected as the default class; a root node is created with 50 instances of this class.
2. The learner is presented with 50 instances of Iris-setosa, all of which are misclassified by the root node as Iris-versicolor.

   A child node of class Iris-setosa is added to the root node. Its classifier, trained on instances of type Iris-versicolor (50) and Iris-setosa (50), is the following decision tree:

   petalwidth $<= 0.6$: Iris-setosa
   petalwidth > 0.6: Iris-versicolor
3. The learner is presented with 50 instances of Iris-virginica, all of which are misclassified by the root node as Iris-versicolor. (The petal width of Iris-virginica is in the range 1.4 to 2.5, thus the Iris-setosa node will not fire on any of the Iris-virginica data.)

   A child node of class Iris-virginica is added to the root node. Its classifier, trained on instances of type Iris-versicolor (50) and Iris-virginica (50), is the following decision tree:

   petalwidth $<= 1.7$
   | petallength $<= 4.9$: Iris-versicolor
   | petallength > 4.9
   | | petalwidth $<= 1.5$: Iris-virginica
   | | petalwidth > 1.5: Iris-versicolor
   petalwidth > 1.7: Iris-virginica

In summary, a node was generated for each class present in the data. The IEL tree consists of a root node with two child nodes. In this case, the IEL hypothesis is semantically identical to a single C4.5 decision tree grown on all data.

5 Learning at Object and Global Levels

5.1 Knowledge Acquisition vs Machine Learning

The proposed system uses knowledge acquisition techniques to build and update object definitions. At a philosophical level, knowledge acquisition may be preferable to machine learning as it excludes the possibility of adopting spurious (but consistent) hypotheses. Knowledge acquisition has been shown also to have practical advantages [Gaines, 1989] especially when access to data is limited.

Ripple Down Rules (RDR) [Compton and Jansen, 1989] is a technique for capturing knowledge directly from human experts using cases and an exception structure. Should the system make an error, an update is made that distinguishes between the misclassified case and the case associated with the rule that made the incorrect classification.

RDR has some advantages over machine learning when learning at the object level:

- Causes of error in identification can be systematically isolated and rectified.
- Updates to object definitions will not alter the performance over previously correctly identified objects.

5.2 Creating an Object Definition

Figure 5 shows the system's graphical interface for new object definition, and for definition update when the system incorrectly identifies an object (refer to section 5.4). The object in question is the beacon shown in figure 3.

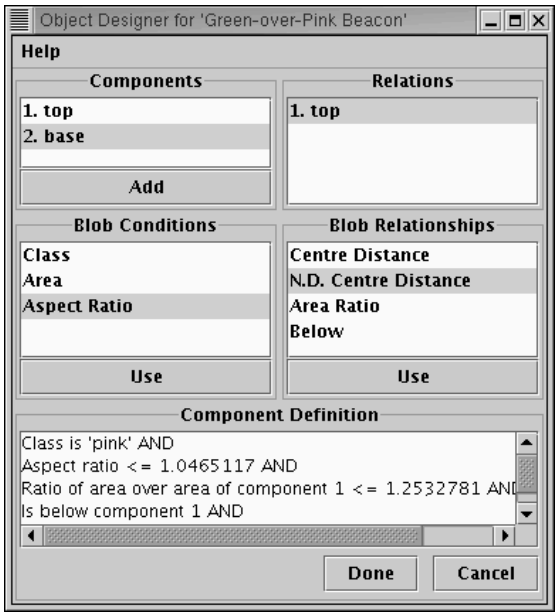

Figure 5: Interface for Object Definition

The definition of an object starts by providing the system with an initial example of the object, and pointing out its salient features. An object is modelled as a set of component blobs. The required properties of each component, and relationships it must satisfy with other components, are defined (and incrementally refined) by the user.

An object's components are entered via the *Components list* (top left). This starts a dialog that asks the user to select a blob, and to label it.

Once a component has been entered, its properties can be used as conditions for the definition. Displayed in the *Blob Conditions list* (lower left) are the subset of blob properties that are applicable to the currently selected component. As visible in figure 5, the system's set of blob properties includes:

- *Class:* the class of the component blob.
- *Area:* the area, measured in number of pixels.
- *Aspect Ratio:* the ratio of the larger dimension over the smaller dimension of the rectangle that bounds the component blob.

The *Relations list* (top right) shows the components with which the selected component in *Components list* may form relationships. This is explained in detail in section 5.3.

If an object consists of two or more components, the relationship between components can also be used as part of the definition. Displayed in the *Blob Relationships list* are the subset of blob relationships that are applicable to the components currently selected in *Components list* and *Relations list*. The set of relationships includes:

- *Centre Distance:* the distance, in pixels, that separates the centres of the rectangles that bound the two blobs.
- *N.D. Centre Distance:* a non-dimensional version of the above relation whereby *Centre Distance* is divided by the square root of the average area of the two blobs.
- *Area Ratio:* the ratio of areas of the blobs.
- *Above:* true if the centre of the rectangle bounding the component blob has a smaller y value than that of the relations blob.
- *Below:* true if the centre of the rectangle bounding the component blob has a larger y value than that of the relations blob.

The selected component's definition is displayed at the base of the interface. In this example, the required properties of the *base* component of a *Green-over-Pink Beacon* are shown.

Once the definition is complete, the system is able to search for instances of the object within input images.

5.3 Object Recognition as a Search Problem

The problem at hand is to take a class image as input (or equivalently an array of blobs) and return a list of objects. Given a model for an object of interest, we solve this problem by searching for blobs that satisfy model constraints.

Take an object, OBJ_1 that consists of three components: C_0, C_1 and C_2. Suppose search proceeds in order of index:

- When searching for C_0, no other components have been located, so candidates blob(s) are selected on blob property criteria only.

  If one or more candidate blobs satisfy the required properties for C_0, the search can continue.

- For each candidate located for C_0:

  Candidates for C_1 are selected on blob property criteria as well as relational requirements with the current candidate for C_0.

  For each non-empty set of candidates for C_1, the search will continue for C_2.

- For each candidate in each set of candidates located for C_1:

  Candidates for C_2 are selected on blob property criteria as well as relational requirements with the current candidates for C_0 and C_1

  Each candidate for C_2 equates to an instance of OBJ_1 being located (all object components have been found).

In the proposed system, this process takes the form of a complete depth-first search, with bias toward testing larger candidate blobs first. Search proceeds in the order in which components were defined by the user. All possible combinations of components that form a defined object will be returned.

5.4 Correction of Object Definitions

The system can make two types of object-recognition error: false positive and false negative. It is up to the user to identify error. Once notified, the system will guide the user through steps to construct an amendment using RDR.

False Positive

False positive entails that the system has wrongly labelled a group of blobs as being an object. The definition of one or more components needs to be made more specific to address this error.

Correction is performed by selecting a misclassified component, then forming an amendment rule using available blob condition(s) and/or relation(s). For example:

- A user defines a green-over-pink beacon as consisting of two components. Component one must be green:

  if (green) then true

  Component two must be pink, and have an aspect ratio less than or equal to 1.03:

  if (pink & $AR \leq 1.03$) then true

- On a subsequent search, the system incorrectly identifies a pink-over-green beacon as being a green-over-pink beacon.
- The user amends the definition of the pink-over-green beacon's pink component with the following exception:

  if (pink & $AR \leq 1.03$) then true except
  if (above green component) then false

False Negative

False negative entails that the system has not located an object that is present in an input image. The definition of one or more components needs to be made more general.

Suppose an instance of a green-over-pink beacon is not identified due to its pink component having an aspect ratio of 1.5. Correction is performed as follows:

- User alerts system that a green-over-pink beacon was missed.
- System asks user to select all components of the missing object.
- The system then presents the user with the condition(s) and/or relation(s) that were not satisfied for each component that was not recognised. Refer to figure 6.
- The user provides an amendment for each non-satisfied condition / relation.
- The following else branch is added to the definition of the pink component:

Figure 6: Interface for Correction of False Negatives

if (pink & $AR = 1.03$) then true except
if (above green component) then false
else if (pink & $AR \leq 1.5$) then true

Notice that the satisfied condition *pink* has also been added to the new rule, to maintain a similar level of generality as the rule that failed to fire. The system would prompt the user as to whether the previously added exception rule should also apply to the new rule.

5.5 Test Results on RoboCup Images

The *Sony Legged League* universe consists of a field, two goals, six beacons for localisation, a ball, and two teams of four robots each. All objects are uniquely colour-coded aside from the robots, with one colour per team.

Testing took place on 40 instances of 8 object types over 22 images. The number of instances of each object ranged from 2 to 8. Table 1 is a summary of system performance[3].

TIMES SEEN	1	2	3	4	5	6	7	8
OBJECT TYPES	8	8	7	5	5	3	3	1
RECOGNISED	0	3	5	4	4	2	3	1
FALSE NEG.	8	5	2	1	1	1	0	0
ERROR (%)	100	63	29	20	20	33	0	0
RULES ADDED	14	7	2	1	1	1	0	0
FALSE POS.	0	3	0	0	0	4	0	0
RULES ADDED	0	3	0	0	0	4	0	0
RULES	14	24	26	27	28	33	33	33

Table 1: System Performance and Growth in Number of Rules used in Object Definitions as Training Progresses

5.6 Discussion

Convergence to Higher Accuracy

Albeit the sample size is small, two promising trends appear to be present in the results:

- The rate of false negatives (objects the system failed to recognise) dropped as corrections were made to object definitions.

[3]Column 1 reads: 'The 1^{st} time objects were seen by the system (one from each of 8 types), none were recognised, and 14 rules were added.' Column 7: 'The 7^{th} time objects were seen by the system (only 3 objects appeared 7 times in test set), all were recognised.'

- The number of corrections required reduced as system experience grew.

These trends suggest that the system learned to recognise user-defined objects by incrementally taking advice on object blob properties and relations.

Example Definition: Pink-over-Blue Beacon

The pink-over-blue beacon's two components were initially defined as follows:

TOP:

if (pink) then (true)

BASE:

if (blue & below(TOP) & CD_{ND}[4] $\leq$ 2.04) then true

The initial definition was found to be overly general, and resulted in three false positive errors. The following updates were made:

TOP:

if (pink) then true except
if (area $\leq$ 3 pixels) then false

BASE:

if (blue & below(TOP) & $CD_N \leq$ 2.04) then true except
if (areaRatio(TOP) $\leq$ 0.00870) then false
else if (areaRatio(TOP) $\leq$ 0.0187) then false

These three updates were required to filter out small blobs of colour that resulted from noise in the camera image and colour classification errors at the pixel level. The user was therefore adding knowledge to compensate for errors at lower levels of the system, as well as providing an object definition.

6 Mixed-Initiative Error Correction

6.1 Problem Setting

Suppose IEL has been employed within SanityCheck to learn to classify pixels in a robot soccer domain. Initially, the user delimits a region containing colours that are not present in any of the objects in this domain. These are the basis of the IEL default node, with class *ignore*. Following this, the user delimits regions of colour that do belong to objects of interest. For each new class, at least one IEL node will be added to the tree.

Assume now that the system can recognise the colour orange, and has an object definition for the ball based on colour and shape only. An image depicting a spectator wearing an orange cap is presented to the system. The user wishes to teach the system that the cap is not a candidate for the ball: this can potentially be done in two ways.

6.2 Pixel-Level Correction

The error could be justified as being due to an overly general definition of orange, that should be made more specific to exclude non-ball objects. Assume that the cap has different spectral properties to previously seen balls, and that this distinction is machine-learnable. By providing the system with the cap's pixels as examples of class *ignore*, the cap blob would cease to exist.

[4]Non-dimensional centre distance. Refer to section 5.2.

6.3 Object-Level Correction

Alternatively, a rule could be added to the definition of the ball that, say, excludes all balls that are further than a threshold distance from the green playing field.

6.4 Are both Applicable?

An object-level correction is possible if there exists adequate language (object properties and relationships) to describe an amendment. It is up to the user to determine if a sensible rule can be added to facilitate this.

Determining whether a useful amendment can be made at the pixel level is more complex. However, the exception-based incremental-learning framework of IEL provides some information that may assist the user in identifying situations where pixel-level correction may fail:

- *New node training error.* Does there exist a spectral difference between ball and cap? High node training error suggests that the classifier employed by IEL cannot separate cap instances from ball instances.
- *New node test error.* Low training error does not imply low error on unseen data. Wide disparity in training and test accuracy is not uncommon, especially when using learners utilising a complete hypothesis space in domains with high dimensional feature spaces.

  If enough data is available, accuracy over a test set can be used to estimate a hypothesis's ability to generalise. Else, the expected test error could be calculated using a statistical evaluation function such as that used by C4.5 for reduced-error pruning [Quinlan, 1993].
- *Complexity of new hypothesis.* Comparing the complexity of the node hypothesis with the number of instances used in its induction and attribute space dimension, and with the complexity of other nodes in the IEL tree, may also provide some insight.
- *Heuristic measures based on tree structure.* Imagine that an IEL tree includes an *orange* node, o_1, and that a *ignore* child node, i_1 is added to correct an orange cap from being mistaken as a ball. Say a new image of a ball is presented to the system, and that some or all of the ball's pixels cause i_1 to fire.

  Should the user correct this by adding an *orange* child node, o_2, to i_1? Did the i_1 amendment make the concept of *orange* too specific, and o_2 is a valid relaxation of this? Or did i_1's classifier remain consistent by learning idiosyncrasies in the training data, rather than a hypothesis to distinguish cap from ball?

  Would our suspicions be further alerted if, at some point in the future, an *ignore* child node, i_2, is added to o_2?

6.5 Correction at Which Level?

Assuming that both options appear plausible, which one should be used? The choice is not always clear as each has its own associated uncertainties. The following issues apply to making correction at the pixel level:

- It has been demonstrated that changes in ambient lighting conditions can seriously affect colour recognition [Hengst *et al.*, 2000]. Making the concept *orange*

more specific may degrade performance in this respect: will the ball be excluded in a future image?

- While fixing the current error in recognition, it does not address situations such as the cap being exactly ball-coloured. What if a spectator were to hold a ball in open view of the robots?

Issues relating to correction via amendment to object definition:

- Amendments are case-based: rule conditions are derived from misclassified object(s) in the current image. Will the amendment be general enough to hold true for other images, or will further amendment rules be required (hence system error and user time) before satisfactory performance is attained?
- How computationally expensive are object-level rules? Some object property and relational tests can be expensive. Correctness may need to be sacrificed for speed in situations of real-time analysis such as robot soccer.

6.6 Dialogs

It is up to the user to decide which path to take when correcting an error in object recognition. The user has access to knowledge, such as the likelihood of an error occurring, or the current and required frame-rate at which a robot processes images, and the computational expense of each amendment, when making a decision.

The system can also offer advice to accompany user knowledge. Given that the user has opted for a pixel-level correction, dialogs may appear to warn of events such as high training error, higher than average hypothesis complexity, or suspicious patterns in an IEL tree.

The aim of system dialogs is to allow a user to make more informed decisions as to whether to adopt a machine-learned amendment at the pixel level, a user-defined amendment at the object level, or no amendment at all.

Conclusions

We have presented a system that learns to identify objects by incorporating knowledge at a number of levels. Pixel-level analysis is performed via supervised learning of spectral and spatial features. The resulting class map is compressed into connected regions of pixels of like class, or *blobs*. Object models are defined by the user as a set of component blobs. Component definitions are also constructed by the user.

The proposed system was tested on images from a robot soccer domain. Test results demonstrate that the system learned to recognise objects by taking advice from users. Knowledge of object properties, as well as remedies for errors passed up from lower levels of analysis, were conveyed to the system.

When SanityCheck is used in conjunction with Incremental Exception Learner, error in object recognition may result in an incremental amendment at either the pixel level (via induction by IEL) or at the object level (via knowledge acquisition using Ripple Down Rules). The user is responsible for selecting the level at which correction occurs, and is assisted in this task by system dialogs.

Acknowledgments

Robot soccer images were provided by rUNSWift, the University of New South Wales RoboCup Sony Legged-League team.

The medical image shown in figure 1 was obtained during the involvement of Julian Kerr in the initial phase of a project being undertaken by the University of New South Wales, Pittwater Radiology Pty Ltd and Philips Medical Systems Australasia Pty Ltd.

This research is supported by the Australian Research Council.

References

[Kerr and Compton, 2002] J.L. Kerr, P.J. Compton. *Interactive Learning when Human and Machine Utilise Different Feature Spaces* The 2002 Pacific Rim Knowledge Acquisition Workshop, 15–29, Tokyo, Japan, Aug. 2002.

[Kerr and Wilson, 2001] J.L. Kerr, P. Wilson. *Web-Based Knowledge Acquisition and Diagnostic Aid in a Medical Imaging Domain.* Knowledge-Based Intelligent Information Engineering Systems and Allied Technologies, 904–909, Osaka, Japan, Sept. 2001.

[Hengst *et al.*, 2000] B. Hengst et al. *The UNSW RoboCup 2000 Sony Legged League Team* RoboCup 2000: Robot Soccer World Cup 4, Springer, 2001.

[Ojala and Pietikainen, 1999] T. Ojala, M. Pietikainen. *Unsupervised texture segmentation using feature distributions* Pattern Recognition **32** 477–486, 1999.

[Uppaluri *et al.*, 1996] R. Uppaluri, T. Mitsa, E. A. Hoffman, G. McLennan, and M. Sonka *Texture analysis of pulmonary parenchyma in normal and emphysematous lung* Medical Imaging 1996: Physiology and Function from Multidimensional Images, Vol. 2709, Newport Beach, CA, pp. 456-467, SPIE, (Bellingham, WA), 1996.

[Quinlan, 1993] J.R. Quinlan. *Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann (1993)

[Gonzales and Woods, 1993] R.C. Gonzales and R.E. Woods. *Digital Image Processing.* Addison-Wesley, 1993.

[Aloimonos, 1990] J. Aloimonos. *Purposive and Qualitative Active Vision.* Proc. of DARPA IU Workshop, 816–828, 1990.

[Gaines, 1989] B.R.Gaines. *An Ounce of Knowledge is Worth a Ton of Data: Quantitative Studies of the Trade-off between Expertise and Data based on Statistically Well-founded Empirical Induction.* International Workshop on Machine Learning **6** 1989.

[Compton and Jansen, 1989] P.J. Compton and R.A. Jansen. *A Philosophical Basis for Knowledge Acquisition.* European Knowledge Acquisition for Knowledge-Based Systems Workshop **3** 75–89, 1989.

[Marr, 1982] D. Marr. *Vision.* Freeman, New York, 1982.