INCREMENTAL LEARNING FOR SEGMENTATION IN MEDICAL IMAGES

Avishkar Misra, Arcot Sowmya, Paul Compton

School of Computer Science and Engineering
University of New South Wales, Australia
{amisra, sowmya, compton}@cse.unsw.edu.au

ABSTRACT

Hand-coded vision systems are problematic in complex medical domains and are hard to change as new information emerges. Knowledge-Engineering and Machine Learning approaches to intelligent vision systems also face learning bottlenecks. We have developed an approach to engineering vision systems, which allowed the user to make incremental changes to refine the performance of the system and address these limitations. A medical image segmentation system was built using this approach. In only a few hours of training, the system was able to exceed the performance of a similar hand-coded system built over a period of three months.

1. INTRODUCTION

Most learning problems for medical image segmentation systems can be formulated into one of three types of learning - Learning (a) algorithm control, (b) tuning of parameters or (c) classification model. All three forms of learning give intelligent vision systems the ability to make prudent processing and interpretation decisions.

Existing approaches to building intelligent vision systems, have had limited success. Knowledge Engineering approaches [1], suffer from a knowledge acquisition bottleneck [2], while Machine Learning approaches [3] cannot learn unless a significant amount of training data is available. In medical domains there is significant inter- and intra-patient variation and often data is limited or only available incrementally.

Previously, we proposed an incremental expert-driven learning framework for vision systems, called *ProcessRDR* [4]. ProcessRDR enables a system to incrementally refine its behaviour with the assistance of an expert, whilst automatically ensuring that knowledge is added in a consistent manner. In this work, we have used ProcessRDR to carry out the three forms of learning required by most segmentation systems, namely algorithm control, tuning of parameters and classification.

The following section describes Ripple Down Rules (RDR), which were used to develop ProcessRDR presented in section 3. In order to evaluate the approach, we have developed a Lung Boundary Extraction (LBE) system capable of delineating the pleura in High Resolution Computed Tomography (HRCT) scans of the lung, and analysed its performance with respect to a hand-crafted vision system in section 4. The results show that ProcessRDR can be used very quickly and easily to develop complex vision systems.

This research was partially supported by the Australian Research Council through a Linkage grant (2002 - 2004), with Medical Imaging Australasia as clinical and Phillips Medical Systems as industrial partners.

2. RIPPLE DOWN RULES

Ripple Down Rules [5] were originally designed to address the knowledge acquisition bottlenecks in non-vision domains. The knowledge is maintained in a nested hierarchy of rules. Each rule defines a *Context* that must be true in order for the rule's *Conclusion* to hold. The inverse hierarchy of rules means that the child rule's conclusion supersedes that of the parent rules. In Figure 1, Rule D is an exception to Rule B. A case satisfying D's context would be classified differently to those cases which only satisfied Rule B's context.

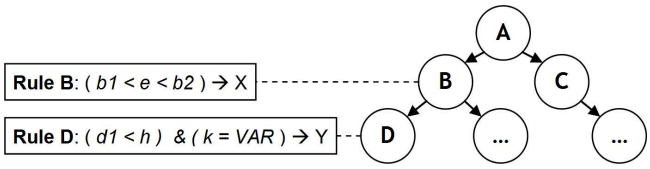


Fig. 1. Ripple Down Rules

The RDR framework allows incremental, ad-hoc refinement of knowledge with only localised knowledge revision. In order to correct a rule, we only need to add an *exception* rule to that rule. As part of this process, any new change is automatically evaluated against existing knowledge to ensure that the change does not affect previously correct knowledge.

RDR have been used for classification problems in vision domains [6] or to incrementally collect labelled data from experts [7] for machine learning. However the benefit of using RDR to learn and manage learning of algorithm control and parameter tuning is presented for the first time in this paper. Work by Beckmann et al [8] in using RDR to control Genetic Algorithms is the closest to our approach.

3. PROCESS-RDR

The ProcessRDR framework brings the benefits of RDR's incremental knowledge acquisition to the construction, maintenance and refinement of vision systems. In ProcessRDR, each configurable processing element within a complex vision system is isolated as a separate learning element, and responsible for learning either algorithm control, parameter values or the classification model. Figure 2 describes the key components of ProcessRDR.

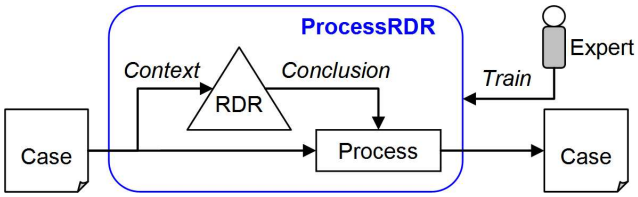


Fig. 2. ProcessRDR Components

In keeping with the RDR terminology, a *Case* is defined as a collection of image and other external information, and is the input to each of the ProcessRDRs. At each ProcessRDR, the case is evaluated and the deepest firing rule's conclusion defines the processing directives to act on the case. The case is routed to various ProcessRDRs for processing until the overall result is achieved.

Each individual ProcessRDR refines its own processing knowledge over time, with the assistance of an expert. If an expert does not agree with the result of a ProcessRDR, they can add an exception rule for that specific ProcessRDR. Alternatively, a simple generalisation scheme ([9]) is used to expand the scope of an existing rule to include a failing case.

3.1. Rule Context

In ProcessRDR, each rule's context is a conjunction of *conditions*. The conditions define measurements to be evaluated on the whole or part of an image, and allow us to make an arbitrary combination of information from multiple processing steps. For example, a region of interest, extracted from one ProcessRDR, can be combined with the output of another ProcessRDR to define a measurement as part of a rule's context. This simple form of information fusion from multiple ProcessRDRs is defined via the following components of a condition.

1. *Scope* defines the region on the image that should be used to measure a particular attribute. The scope can be defined as *Global*, in which case the measurement is made on the entire image. Alternatively it can be on a smaller *Region of Interest* (ROI), bounded by a polygon. This allows us to encode local context-specific knowledge.
2. *Measure Name* describes the particular attribute to be measured on the case. These measurements fall into one of the following categories:
   - *Image Measures* defines numerical and statistical measures on image pixel values. For example, mean and standard deviation of image intensities, or the Gray Level Co-occurrence Matrix describing the texture.
   - *Processing History* describes the given case's processing history. For example, the *Previous Processing Step* or *Number of processing steps*. These forms of measures help evaluate Control knowledge for each case.
   - *External Information* evaluates external domain-specific information. For example, Patient Information or Image modality-specific information.
   - *Structural Measures* are numerical measures on the shape properties of extracted regions in the image, delineated by a bounded polygon. For example, area, perimeter, moments, etc.
3. *Evaluation* describes the type of evaluation condition, such as $=, \leq, \geq, \neq$ *or* $\in$.
4. *Value* is a numerical or nominal value against which the measure and equality must be evaluated.
5. *On Image* directive allows a rule to refer to historical information, thereby combining information from multiple ProcessRDRs. For example, OnImage value of "Thresholder" specifies that the measurement should be made on the result of Thresholder ProcessRDR.

Consider the following sample rule condition: $roi : image : intensity : mean :<: 50 : OnImage = Original$; The condition asks the system to evaluate the mean image pixel intensity on a region on the Original image, defined by the ROI. The condition will be true if that value is less-than 50.

3.2. Rule Conclusions

The conclusions of a ProcessRDR can take three possible forms, depending on the type of learning necessary. These include:

- *Control Directives* - This form of conclusions defines a sequence of processing subtasks or algorithms to run on a given case. For example, in order to remove noise, a morphology ProcessRDR would choose different sequences of morphology operations under different circumstances. For example, sequence A may be **(dilation, erosion)**, whilst sequence B is **(medium-filter, medium-filter)**. Alternatively, the Control Directive may invoke another ProcessRDR or decide to stop processing altogether.

- *Parameter Values* - This form of conclusion defines the parameter values to be used for a specific processing task. For example, a Thresholding ProcessRDR would define the *minimum* and *maximum* threshold values to use to binarize a gray-scale image.

- *Classification* - This form of ProcessRDR defines how a potential input case should be labelled. For example a Lung Labelling ProcessRDR assigns a label to each of the extracted closed regions as lung or not-lung. The conclusions of this ProcessRDR are control directives.

- *Lung Selector ProcessRDR* - is responsible for accepting the extracted region outlines as lung or rejecting them as external noise. This ProcessRDR has a conclusion type of classification, so the ProcessRDR incrementally constructs a lung classification model.

4. LUNG BOUNDARY EXTRACTION USING PROCESS-RDR

In order to evaluate the effectiveness of such a framework we applied the technique to lung boundary delineation in High Resolution Computed Tomography (HRCT) scans. Lung Boundary Extraction (LBE) is a difficult process due to the high level of variation present within a patient's HRCT study and across patients. The lung boundary delineation is made even more difficult with the presence of motion artifacts or diseases near the lung periphery. Fixed algorithms without the ability of contextually specific processing often fail.

4.1. Experiment Setup

A non-adapting hand-crafted system called *LBE-Fixed* was compared against a ProcessRDR based system. LBE-Fixed is an unpublished tool developed in our group and used in a range of research. The *LBE-ProcessRDR* system is composed of the following elements:

- *Control ProcessRDR* - designed to learn the overall processing control knowledge associated with lung boundary extraction. This ProcessRDR describes which of the other ProcessRDRs to invoke on a case given certain conditions. The conclusions are control directives and so carry out algorithm control learning.

- *Thresholder ProcessRDR* - responsible for appropriate binarization of a gray-scale image, to maximise the differentiation between regions inside the lung and outside the lung. Selection of good threshold values can help avoid motion artifacts (due to the heart) and deal with noisy images as a consequence of diseases that might be present. The conclusions are parameter values for Thresholding and so carry out parameter tuning.

- *Outliner ProcessRDR* - is responsible for cleaning up the result from a binarization process, by making prudent decisions on the sequence of morphology operators to apply. In addition to this, the ProcessRDR allows the system to use different outlining algorithms under differing circumstances. Here tactical knowledge is captured by the rules created by the vision expert.

The 18 patient studies were randomly distributed into training and test sets. Each image within the training set was processed by both LBE-Fixed and LBE-ProcessRDR. In the event that LBE-ProcessRDR failed, a vision expert would train one or more ProcessRDRs for that case. A *failure* occurs, if the system fails to extract the lung regions correctly, or if it incorrectly includes non-lung regions as part of the lung.

Note that each of the ProcessRDRs were initialised with no prior knowledge. An example of results from individual ProcessRDRs is shown in Figure 4, together with the original image with extracted lung regions superimposed on it.

4.2. Results

The graph in Figure 3 shows the failure rates of both LBE-Fixed and LBE-ProcessRDR during training. Since LBE-ProcessRDR is trained for each failure, the number of rules corrected for that case is also shown as a histogram on the same graph (right y-axis). LBE-ProcessRDR's adaptive nature has meant that the system does not repeat the same errors in future. After the 14 training studies, the LBE systems were evaluated on 4 unseen studies (74 images). The LBE-ProcessRDR was not corrected during evaluation. As shown in Table 1, LBE-ProcessRDR had half as many errors as LBE-Fixed.

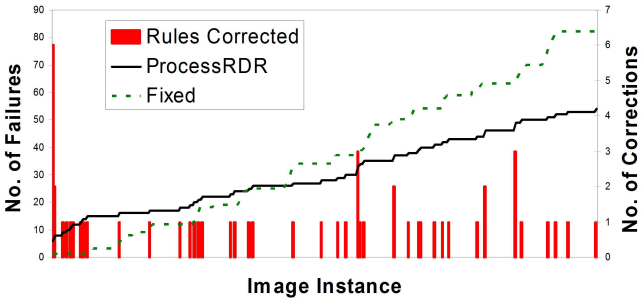


Fig. 3. Training Phase Failures

The time-stamp information in the rules allowed us to accurately analyse the amount of time taken to carry out training for LBE-ProcessRDR. Discounting for coffee breaks, the time taken to process and train the 14 studies was only 4 hours and 31 minutes. It is quite impressive when we consider that the LBE-Fixed was manually crafted by a vision expert over 3 months. During training 29 exceptions and 25 generalisations were carried out across the 4 individual ProcessRDRs.

Table 1. Failure Summary

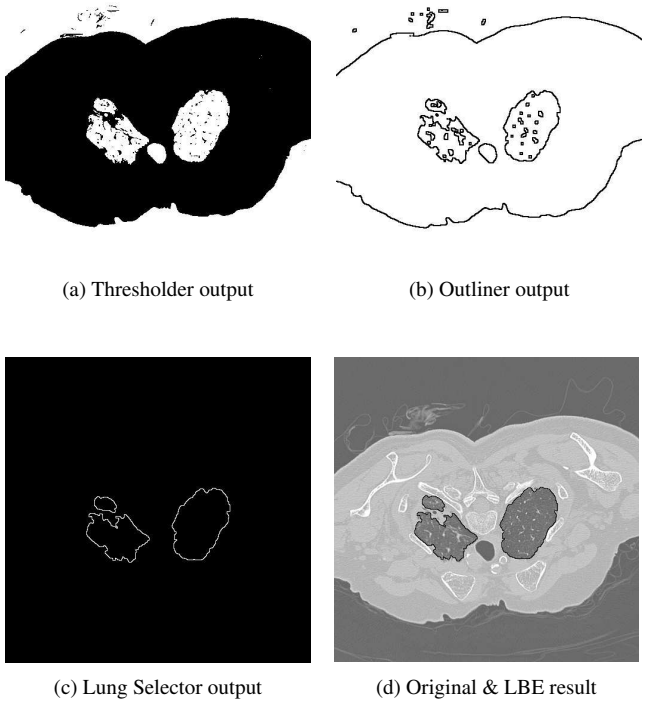
Dataset	Patients/Images	LBE Failures (%)	
		Fixed	ProcessRDR
Train A	7/141	13	10
Train B	7/129	17	10
Test	4/74	14	6

5. CONCLUSIONS

As shown by LBE-ProcessRDR, a collection of ProcessRDRs with no prior knowledge quickly learned to match and exceed the performance of a fixed algorithm system handcrafted by a vision expert. It confirms the intuition that an adaptive system will outperform a non-adaptive system over a significant period of time. Though adaptive systems may not converge to zero errors, it does allow the satisfaction of correcting any errors of concern. We have shown that ProcessRDR can be used to do so easily. The ProcessRDR framework further demonstrates the ease with which such adaptive system can be developed.

ProcessRDR is able to manage the complexity associated with resolving conflicts and contradictions of ad-hoc development in complex vision systems. A ProcessRDR element's Conclusion applies to a case only under a specific Context. This ensures that a solution which is correct for existing cases, continues to operate correctly for those cases, in the event of a change to handle exceptional cases.

ProcessRDR can deal with exceptional cases that Machine Learning would find difficult to accommodate, whilst learning algorithm control, parameter values and classification model. Presently we are using ProcessRDR to extract and develop complete lung anatomy. Also, our current work in local theory revision for ProcessRDR will minimise knowledge fragmentation and help inexperienced experts in rule creation.



(a) Thresholder output (b) Outliner output

(c) Lung Selector output (d) Original & LBE result

Fig. 4. LBE-ProcessRDR Outputs

6. REFERENCES

[1] R. Clouard, A. Elmoataz, C. Porquet, and M. Revenu, "Borg: A knowledge-based system for automatic generation of image processing programs," in *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 21, No. 2*, 1999, pp. 128–144.

[2] B. Draper, A. Hanson, and E. Riseman, "Knowledge-directed vision: control, learning and integration," in *Proc. of IEEE, vol. 84, n. 11, 1996.*, 1996, pp. 1625–1681.

[3] B. Bhanu and J. Peng, "Adaptive integrated image segmentation and object recognition," in *IEEE Trans. on Systems, Man and Cybernetics, Vol. 30, No. 4*, 2000, pp. 427–441.

[4] A. Misra, A. Sowmya, and P. Compton, "Incremental learning of control knowledge for lung boundary extraction," in *Pacific Knowledge Acquisition Workshop, Auckland, New Zealand*, 2004.

[5] P. Compton, G. Edwards, B. Kang, L. Lazarus, R. Malor, T. Menzies, A. Srinivasan P. Preston, and C. Sammut, "Ripple down rules: Possibilities and limitations," in *6th Banff AAAI Knowledge Acquisition for Knowledge Based Systems Workshop, Banff, 6.1 - 6.18*, 1991.

[6] P. Singh and P. Compton, "Evolution oriented semi-supervised approach for segmentation of medical images," in *Proceedings of ICISIP 2004, India*, 2004, pp. 77–81.

[7] J. Kerr and P. Compton, "Toward generic model-based object recognition by knowledge acquisition and machine learning," in *Proceedings of the IJCAI-2003 Workshop on Mixed-Initiative Intelligent Systems. Acapulco*, 2003, pp. 80–86.

[8] J. Bekmann and A. Hoffmann, "Incremental knowledge acquisition for improving probabalistic search algorithms," in *Engineering in the Age of the Semantic Web, Germany*, 2004, pp. 248–264.

[9] T. Scheffer, "Algebraic foundation and improved methods of induction of ripple down rules," in *Proceedings of the Pacific Rim Workshop on Knowledge Acquisition. Sydney, Australia*, 1996.